UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Aplicación web para la gestión completa de los TFG “TFGTool”

Óscar Ruiz Rodríguez

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Aplicación web para la gestión completa de los TFG "TFGTool"

Autor: Óscar Ruiz Rodríguez
Tutor: Eloy Anguiano

julio 2020

Óscar Ruiz Rodríguez
Aplicación web para la gestión completa de los TFG "TFGTool"

Óscar Ruiz Rodríguez
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

Me gustaría agradecer el apoyo y la ayuda que he recibido por parte de mi tutor Eloy Anguiano. Agradecer la oportunidad que me ha brindado ofrenciendome esta propuesta y también la orientación y consejos que me ha ido brindando en las últimas etapas del grado y en este trabajo para lograr terminar con éxito el TFG.

Me gustaría agradecer también a “Grapastagoras” por todo el trayecto recorrido desde que empecé en la carrera. Hay conocimientos que no se adquieren estudiando y la persona que soy hoy en día en parte es gracias al tiempo y las aventuras que he pasado con vosotros dentro y fuera de la facultad.

A Jimena, por haber sido un hombro en el que apoyarse dentro y fuera de la facultad y una gran compañera.

A Gloria, compañera de mil batallas. Una persona muy especial que apareció de la nada en aquella clase de *Calculo I*, pero que desde ese momento ya sabia que iba a ser una amiga más allá del grado.

No me puedo olvidar de Alfonso, mi mejor descubrimiento dentro de la carrera. Has sido una persona clave para seguir hasta el final y te has convertido en un compañero de esos de vida, que valen más que un título.

Y por último, y por ello más importante, a mis padres. Sin su apoyo, sin su consejo, sin su guía, estas palabras nunca se hubieran escrito. Soy quien soy gracias a ellos. Gracias por entender y soportar todos los baches que he tenido y por haber sido ese hogar que reconfortaba y daba sentido a toda lucha.

Resumen

Dado los tiempos en los que vivimos, se ha tornado una necesidad disponer de herramientas que nos provean de los servicios necesarios para poder desarrollarnos con el máximo potencial. Durante los últimos años, las universidades y el conjunto de la sociedad ha asimilado la importancia de los recursos que nos provee la web.

Es por esto, que la universidad, ha visto necesario la creación de un portal, en el que estudiantes y profesores puedan compartir y automatizar tareas durante el desarrollo de los trabajos de fin de grado (TFG). Gracias a este portal, los estudiantes podrán seleccionar y trabajar en un TFG del conjunto total de trabajo que estarán accesibles en el portal y los profesores podrán compartir el estado y comprobar el desarrollo de estos.

Además la gestión de defensa, la asignación de la nota y el cierre de actas se completaran con esta aplicación.

Palabras clave

TFG, Universidad, Aplicacion, Web, Desarrollo, Requisito, Usuario, Estudiante, Tribunal, Estados, Solicitud, Admitido, Defendido, TFGTool

Abstract

Given the times in which we live, there has become a necessary have tools that provide us the necessary services in order to develop with the maximum potential. In recent years, universities and society as a whole have assimilated the importance of the resources provided by the web.

For this reason, the university has seen the creation of a portal necessary, in which students and professors can share and automate tasks during the development of the final degree projects. Thanks to this portal, students will be able to select and work on a TFG from the total set of work that they will be able to access on the portal, and teachers will be able to share their status and check their development.

In addition, defense management, assignment of the note and closing of proceedings is completed with this application.

Keywords

TFG, University, Application, Web, Development, Requirement, User, Student, Court, States, Application, Admitted, Defended, TFGTool

Índice

1 Introducción **1**

1.1 Objetivo . . . 1

1.2 Motivación . . . 1

1.3 Organización de la memoria . . . 1

2 Estado del Arte **3**

2.1 Introducción . . . 3

2.2 Actualidad . . . 3

2.3 Otras Plataformas . . . 4

2.4 Metodología . . . 6

2.5 Tecnologías . . . 7

3 Diseño **11**

3.1 Análisis de requisitos . . . 11

3.2 Casos de uso . . . 13

3.2.1 Estudiante . . . 13

3.2.2 Profesor . . . 13

3.2.3 Tribunal . . . 15

3.2.4 Administración . . . 16

3.2.5 Casos de uso detallado . . . 16

3.3 Diagrama de flujo . . . 18

3.3.1 Solicitud TFG Estudiante . . . 18

3.3.2 Admitir TFG Profesor . . . 19

3.3.3 Seleccionar Tribunal . . . 19

3.3.4 Admitir TFG Administración . . . 21

3.3.5 Admitir TFG Tribunal . . . 21

3.3.6 Evaluar TFG Tribunal . . . 22

3.4 Diseño Base de Datos . . . 23

4 Desarrollo **25**

4.1 Metodología Agile aplicada al proyecto . . . 25

4.2 Equipo de trabajo . . . 25

4.2.1 Product Owner . . . 25

4.2.2 Scrum Master . . . 26

4.2.3 Equipo de Desarrollo 26
4.3 Sprints 26
4.3.1 Solicitud 26
4.3.2 Admitido 29
4.3.3 Evaluado 31
4.3.4 Calificado 33
5 Integración, Pruebas y Resultados **35**
5.1 Pruebas con Cliente 35
5.2 Pruebas de Sistema 35
5.2.1 Solicitud 35
5.2.2 Lectura 36
5.2.3 Admitido 36
5.2.4 Tribunal 36
5.2.5 Evaluado 36
5.2.6 Calificado 37
5.3 Pruebas de Usuario 37
5.4 Pruebas de Componentes 38
5.5 Pruebas de Compatibilidad 40
6 Conclusiones **41**
6.1 Conclusión 41
6.2 Trabajo futuro 42
Bibliografía **43**

Listas

Lista de códigos

4.1 Solicitud Defensa. 28
4.2 Pendiente Defensa. 28
4.3 Admitir Administracion. 30
4.4 Crear Tribunal. 31
4.5 Admitir Administración. 32
4.6 calificado. 34

Lista de figuras

2.1 Dashboard de la aplicación 3
2.2 Aspecto de TFG de la aplicación 4
2.3 Ejemplo de uso de figure 5
2.4 Metodología Scrum 6

3.1 Casos de uso Alumno 14
3.2 Casos de uso Profesor 14
3.3 Casos de uso Tribunal 15
3.4 Casos de uso Administración 16
3.5 Flujo solicitud estudiante 19
3.6 Flujo admitir profesor 20
3.7 Flujo integrantes tribunal 20
3.8 Flujo admitir administración 21
3.9 Flujo evaluar tribunal 22
3.10 Flujo admitir TFG 22
3.11 Esquema relacional base de datos 24

4.1 Asignado a Defensa solicitada 27
4.2 Defensa solicitada a Pendiente defensa 29
4.3 Pendiente defensa a Tribunal asignado 30
4.4 Tribunal asignado a Defendido 32
4.5 Defendido a Calificado 33

5.1 Mensaje error componentes 38

5.2 Ejemplo navegadores 40

Lista de tablas

5.1 Tabla de tiempos de logro 38
5.2 Tabla de ejemplo 40

1

Introducción

1.1. Objetivo

El obetivo de este TFG es el de desarrollar una aplicación web que permita a todos los usuarios ser participes de la tramitación de su propio TFG, además de permitir la gestión de estos a profesores, tribunal y administración y simplificar los procesos administrativos asociaciodos a los TFG. Por otro lado, el fin de este trabajo es mejorar los conocimiento y la capacidad de autodeterminación gestionando un proyecto.

1.2. Motivación

El desarrollo de este Trabajo de Fin de Grado pretende evolucionar el trabajo previo que ya existía. Por el momento y antes de llevar este a cabo este proyecto, los usuarios tenían la capacidad de listar los TFG disponibles, y de conocer, parcialmente, el estado de ellos. Esta funcionalidad no completaba del todo los requerimientos y el potencial de esta herramienta. Es por ello que surgió este proyecto, gracias a este desarrollo todos los usuarios de la aplicación podrán observar y ser participes de la evolución de los TFG que se realicen en la Universidad Autónoma de Madrid (UAM).

1.3. Organización de la memoria

La memoria se ha estructurado en seis capítulos distintos donde queda recogido todo el conocimiento que se quiere plasmar.

El primer capítulo es introductorio y en el se puede observar que adelantará el tema principal de la memoria del proyecto. Este primer capítulo está seguido de otro subíndice llamado “Motivacion” donde se plasma el fin por el cual este trabajo se ha desarrollado.

En el segundo capítulo se explica el estado del arte, donde se realizará un análisis del estado de la tecnología antes del inicio de este trabajo y se comparará con otras universidades para analizar el estado del mercado. Por otro lado, abordaremos la tecnología implicada en este trabajo y la metodología de desarrollo empleada.

En el capítulo tres, se mostrará el diseño adoptado para la realización de la aplicación detallando los requisitos que se han acordado y cumplido, y mostrando a su vez, de forma gráfica como la aplicación esta diseñada para proporcionar la solución.

Después del diseño, en la memoria se encontrará el capítulo cuatro, que está centrado en el desarrollo de la aplicación. En el se detalla como el equipo estaba organizado y cuales eran los papeles que tenían cada uno. Además los avances que se han realizado en función del *sprint* de forma minuciosa y gráfica.

En el penúltimo capítulo se habla de la integración y como fue esta durante el *sprint* correspondiente. Además se muestra una tabla con información relevante de usuarios que han probado la aplicación y se plasma un conjunto de imágenes que muestran cual es el comportamiento de la aplicación ante un funcionamiento anómalo.

Por ultimo, en el sexto capítulo se encuentran las conclusiones, donde y tras acabar todo el trabajo, se define cuáles han sido las conclusiones de desarrollar el Trabajo Fin de Grado, además de mostrar las posibles pautas a seguir para mejorar y evolucionar la aplicación.

2

Estado del Arte

2.1. Introducción

En la actualidad, la realización de los TFG es una labor obligatoria para todos los estudiantes que quieran formalizar la finalización de un grado universitario, donde deben poner a prueba las habilidades y los conocimientos adquiridos en el grado que han cursado y al mismo tiempo mostrar las capacidades de síntesis y redacción. Estos trabajos, originalmente se gestionaban de forma presencial en algún paso de todo el proceso comprendido entre la solicitud y la defensa del mismo, pero eso esta apunto de cambiar.

2.2. Actualidad

A día de hoy, la UAM dispone ya de una plataforma básica de gestión de TFG. Siguiendo la definición que la propia web indica, es una herramienta “Para la operatividad administrativa necesaria para los Trabajos de Fin de Grado” [1].

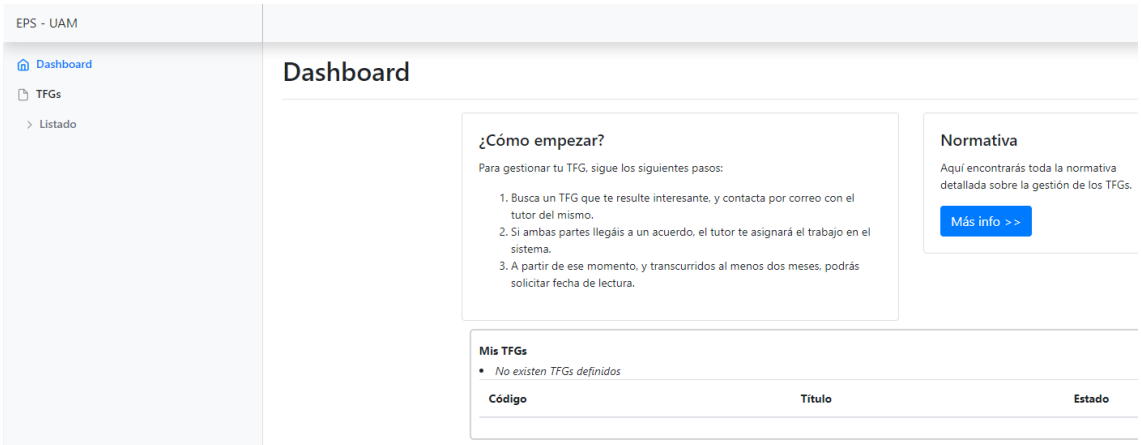


Figura 2.1: Vista original del dashboard de la herramienta TFGTool

Como se puede ver en la figura 2.1 podemos observar el menú principal del portal. En este apartado los usuarios pueden ver cual es el TFG en el que están trabajando. Por otro lado, como la figura 2.2 muestra, la aplicación permite obtener un listado de todos los TFG que están disponibles o asignados.

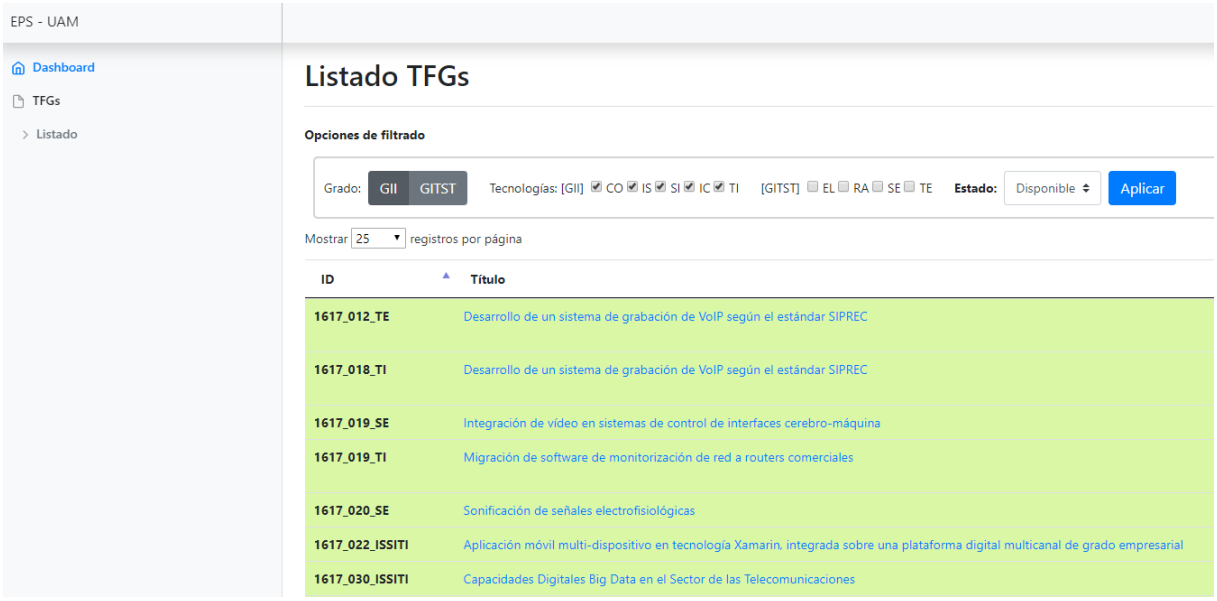
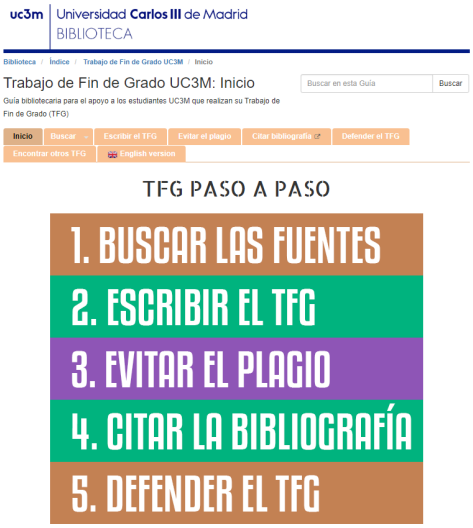


Figura 2.2: Vista original del listado de la herramienta TFGTool. En este listado se proporciona al usuario la opción de entrar en el detalle de un TFG al seleccionarlo

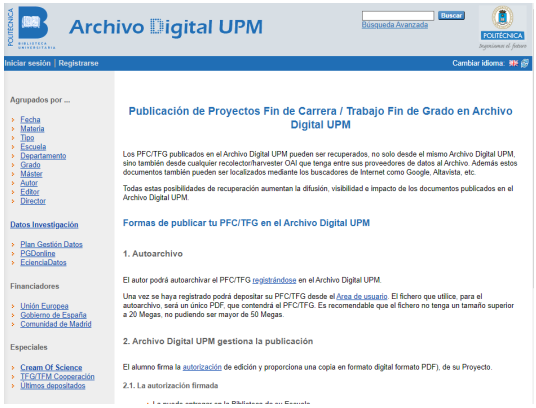
2.3. Otras Plataformas

Esta plataforma no es exclusiva sólo de la Universidad Autónoma de Madrid. Otras universidades han implementado una solución, permitiendo de esta forma mejorar la gestión de dichos trabajos. Tras comprobar las aplicaciones desarrolladas por otras universidades españolas (muestra de la interfaz gráfica de las paginas principales en la figura 5.1), se puede comprobar que todas ellas disponen de portales donde los estudiantes pueden obtener información relacionada con los TFG.

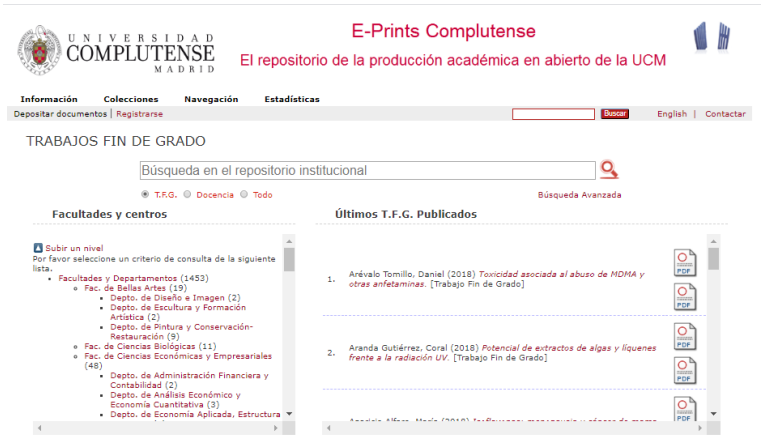
Tras un análisis, se puede comprobar que cada una de ellas ofrecen herramientas que facilitan la gestión en el desarrollo del trabajo, pero todas ellas entre si, tienen diferencias y ninguna aporta una gestión completa para el desarrollo de un TFG.



(a) Aspecto de la plataforma de la Universidad Carlos Tercero de Madrid



(b) Aspecto de la plataforma de la Universidad Politécnica de Madrid



(c) Aspecto de la plataforma de la Universidad Complutense de Madrid

Figura 2.3:

2.4. Metodología

Le metodología usada en este desarrollo es la ya ampliamente conocida, metodología **Scrum** [2]. Este tipo de metodología esta recogida dentro del marco de las Metodologías Ágiles. Estas metodologías son usadas para mejorar la eficiencia y eficacia del desarrollo de tareas y se fundamentan en un desarrollo iterativo de partición de las tareas, en tareas más simples, junto con la comunicación de grupos de trabajo.

SCRUM

La metodología **Scrum** vista en la figura 2.4 es un método pensado para trabajar en iteraciones o *"sprints"*. Estas iteraciones se planifican en un lapso de unas semanas, donde se deben realizar la tareas establecidas. A continuación se exponen las principales características de esta metodología.

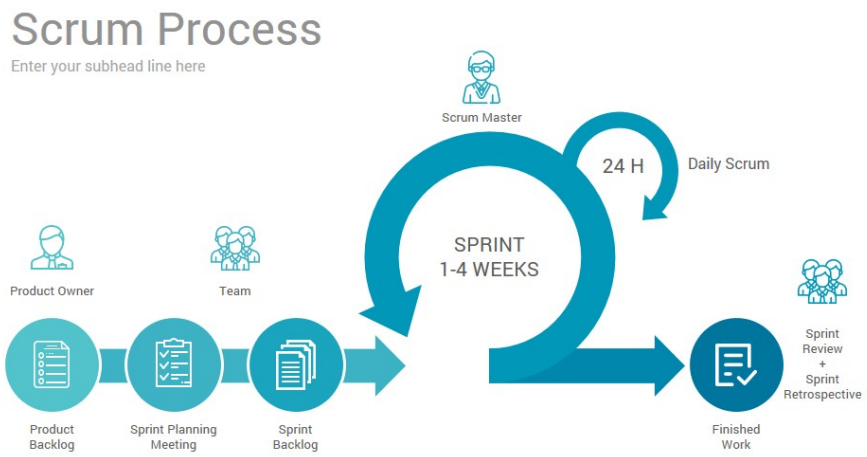


Figura 2.4: Metodología **Scrum**. Figura explicativa donde se ven los actores implicados dentro del proceso. Además de las fases que existen dentro del total del *sprint* [3]

- Los lapsos de tiempo suelen estar comprendidos entre 1 y 4 semanas y comienzan cuando finaliza el anterior.
- Están formados por "Dayly Meeting", reuniones breves diarias donde todos los actores se reúnen para discutir el estado actual del proyecto.
- "Project Owner" que es el encargado de asegurar que el desarrollo del proyecto sigue la normas de la estrategia de negocio.
- "Scrum Master", que es la persona encargada de vigilar que el desarrollo de las tareas siguen los tiempo y los protocolos establecidos.
- "Developer Team" que como su propio nombre indica, es el equipo de desarrollo que se encarga de realizar las tareas impuestas en estos *sprints*.

2.5. Tecnologías

TFGTool es una herramienta web y como tal se nutre de las tecnologías y las metodologías. Dado que el desarrollo de esta aplicación se fundamenta en entorno web, la metodología de desarrollo escogida es **UML-Based Web (UWE)** [4].

Al depender de un modelo de datos, dada la lógica escogida para desarrollar la aplicación, disponemos de un motor de base de datos SQL (**SQLite**) [5] usado para almacenar los datos necesarios para el desarrollo correcto de la aplicación.

Toda aplicación web necesita de un *framework* que facilite la ejecución de la aplicación, además de dar soporte y alivie la carga de computación. En esta aplicación el *framework* usado en esta aplicación es **Cherrypy** y es el indicado por su eficiencia, ligereza y por estar orientado al desarrollo de aplicaciones web basado en objetos con lenguaje **Python**.

Como no podía ser de otro modo, el lenguaje empleado en el núcleo del programa es **Python** usado por su gran versatilidad, potencia y extensión. Para el desarrollo de la parte visual "front-end" el lenguaje usado es **JavaScript**.

SQLite

SQLite es el motor de base de datos que se ha usado para esta aplicación. Las ventajas para usar este cliente son las siguientes:

Ligereza: La biblioteca que contiene este cliente pesa 600 kB.

Versatilidad: Soporta base de datos de terabytes de tamaño.

Portabilidad: No necesita servidor para operar, la base de datos se integra con la aplicación y se almacena en un fichero.

Transaccional: *Atomicity, Consistency, Isolation and Durability*. Cumple cada una de estas características, que son las necesarias para que se lleve a cabo una transacción.

- **Atomcidad:** La atomicidad nos asegura que la transacción se realiza por completo. Si se produce algún fallo, la transición de candela y se vuelve al estado anterior.
- **Consistencia:** "La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido" - Wikipedia [6].
- **Aislamiento:** El aislamiento nos asegura que si una o mas transacciones se están ejecutando al mismo tiempo, no tendrán el conocimiento del estado intermedio de las transacciones.

Durabilidad: "Esta propiedad asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer aunque falle el sistema y que de esta forma los datos sobrevivan de alguna manera" - Wikipedia [7].

Zero-configuration: No requiere de configuración inicial.

CherryPy

El *framework* que se ha utilizado para el desarrollo de la aplicación web es "Cherrypy" [8].

El uso de este *framework* [9] es el adecuado para esta aplicación. Al código fuente de la aplicación esta desarrollado en python por lo que este *framework* es el ideal ya que esta orientado a objetos en Python.

Entre sus ventajas se encuentra el hecho de que es un *framework* simple, modular, extensible.

PYTHON

El núcleo del programa esta desarrollado en Python. Este lenguaje es ampliamente conocido y usado en la actualidad, y no es de extrañar, gracias a una extensa lista de ventajas que lo hacen muy competitivo e interesante de usar. Python es un lenguaje de programación interpretado, multiparadigma, multiplataforma, de código abierto y ampliamente soportado por la comunidad.

Interpretado: A diferencia de los compilados, estos necesitan de un interprete para funcionar. La diferencia con respecto a un lenguaje compilado es la rapidez, siendo los compilados mas rápidos que los interpretados. Por otro lado los lenguajes interpretados son mas flexibles permitiendo un tiempo de desarrollo mas rápido [10].

Multiparadigma: Este lenguaje permite mas de un paradigma de codificación [11], eso quiere decir que se puede usar mas de un método de estructuración y organización de un programa

Multiplataforma: Esta característica permite operar al lenguaje en múltiples plataforma informáticas sin necesidad de una compilación especifica para cada sistema [12].

Código Abierto: Accesible para el total de la comunidad y es libre de ser modificado.

JavaScript

JavaScript es un lenguaje de programación interpretado, ampliamente utilizado y su uso característico es en la parte de cliente. Este lenguaje mejora la interacción con las paginas web ya que aporta dinamicidad a los documentos HTML.

Multiplataforma: Este lenguaje permite mas de un paradigma de codificación, eso quiere decir que se puede usar mas de un método de estructuración y organización de un programa.

Imperativo y estructurado: Este tipo de lenguaje es el paradigma de programación mas antiguo. Las instrucciones se ejecutan unas tras otras.

Orientado a objetos: La programación orientada a objetos crea entidades que se relacionan entre si, permitiendo el tratamiento de los datos de una oferta mas orientativa.

3

Diseño

La aplicación que se ha desarrollado en este TFG es una aplicación que ya esta en funcionamiento, por lo que el diseño ha estado condicionado a los requerimientos originales. El diseño, debido a la metodología escogida, se ha realizado mediante fases, en la que en cada reunión se definían nuevos casos y funcionalidades a implementar además de comprobar y corregir los errores del desarrollo y del concepto. A continuación se describe con más detalle.

3.1. Análisis de requisitos

En este punto se va a exponer de forma individual y detallada el conjunto total de requisitos funcionales que componen el sistema. Estos requisitos están divididos en dos: requisitos funcionales, que son aquellos requisitos que completan la funcionalidad pedida por el cliente y están relacionado con el flujo de la aplicación; y los requisitos no funcionales, que son los requisitos que están relacionados con todo aquello que no esta estrictamente relacionado con la lógica de la aplicación, como el rendimiento, la usabilidad, la mantenibilidad... que determinan el comportamiento de la aplicación [13].

Requisitos funcionales

RF-1.– Login.

- **RF-1.1.– Estudiantes**. Los estudiantes podrán acceder a la aplicación del mismo modo que lo realizaban hasta el momento.
- **RF-1.2.– Tutores**. Al igual que los estudiantes, estos podrán loguearse y usar la funcionalidad pensaba para ellos.
- **RF-1.3.– Tribunal**. El tribunal, formado por profesores, tendrán una funcionalidad especifica.
- **RF-1.4.– Administración**. Este usuario es nuevo en la aplicación, realizara las tareas pensadas para el departamento de administración.

RF-2.– Listar TFG. Todos los usuarios podrán listar el total de los trabajos que se encuentran en la aplicación.

RF-3.– Crear propuesta TFG.

- **RF-3.1.– Profesores**. Los profesores serán los únicos usuarios capaces de crear un TFG .

RF-4.– Seleccionar TFG.

RF-4.1.– Estudiantes. Los estudiantes podrán seleccionar el TFG que tienen asignado y depende del estado podrán realizar distintas funciones

RF-4.2.– Profesores. Los profesores al igual que los estudiantes podrán seleccionar los TFG y dispondrán de distintas funcionalidades dependiendo del estado.

RF-4.3.– Tribunal. El tribunal podrá seleccionar el TFG y acceder a el.

RF-5.– Solicitar defensa TFG. Funcionalidad única del estudiante, mediante la cual el estudiante accederá a la ventana que le permitirá subir su TFG

RF-6.– Descargar normativa. Esta funcionalidad descarga la normativa relacionada de los TFG.

RF-7.– Examinar TFG. Mediante esta funcionalidad el estudiante puede cargar el TFG que quiere subir.

RF-8.– Almacenar TFG. Tras cargar el TFG que quiere enviar, el estudiante deberá presionar sobre el boton para que el TFG se almacene en el sistema

RF-9.– Generar informe TFG.

RF-9.1.– Profesores. En caso de los profesores, estos redactaran un informe en el estado el cual el TFG se prepara para ser aceptado por el tribunal.

RF-9.2.– Tribunal. El tribunal escribirá sus propias anotaciones en la fase de evaluación del TFG.

RF-10.– Admitir TFG.

RF-10.1.– Tribunal. El tribunal sera uno de los actores encargados de admitir el tribunal para evaluación.

RF-10.2.– Administración. La administración es el otro usuario capaz de admitir un TFG para que continúe su evaluación.

RF-11.– Hacer Publico TFG. Mediante esta funcionalidad el tribunal permite que un TFG sea de carácter publico y este se publique.

RF-12.– Crear Tribunal TFG. Mediante esta funcionalidad se deciden los profesores que formaran parte del tribunal.

RF-13.– Acceder informe de tutor. Los profesores del tribunal tienen la capacidad de descargar el informe realizado por el tutor en la fase previo al inicio de la defensa del TFG.

RF-14.– Descargar TFG. Los profesores pertenecientes al tribunal, mediante la pantalla de evaluación pueden descargarse el TFG del estudiante.

RF-15.– Evaluar TFG. Tras asistir a la defensa del proyecto, los participes del tribunal pueden escribir sus anotaciones y dar la nota al trabajo.

RF-16.– Calificar TFG. Tras obtener la evaluación de todos los participantes del tribunal, el presidente es el encargado de generar la calificación del trabajo.

Requisitos no funcionales

RNF-1.– Usabilidad. La aplicación debe ser de uso sencillo

RNF-2.– Rendimiento. La aplicación esta sujeta a requerimientos de rendimiento para su aceptación.

RNF-3.– Portabilidad. La aplicación puede ser usada en distintos navegadores y dispositivos.

RNF-4.– Integridad. La información proporcionada y usada ha de ser cuidada para protegerse contra la corrupción y estados inconsistentes.

RNF-5.– Confidencialidad. El acceso a la aplicación estará garantizado para los usuarios registrados en el sistema.

RNF-6.– Disponibilidad. Se garantiza el acceso a los usuarios.

RNF-7.– Interfaz. Mantiene el estándar de la organización.

3.2. Casos de uso

3.2.1. Estudiante

Este es el caso de uso relacionado con el Estudiantes. Como se puede ver en la imagen figura 3.1, el estudiante tiene una serie de funcionalidades que se corresponden con un flujo normal con respecto a la gestión del TFG entre los que se encuentra “Solicitar defensa TFG”, “Explorar ficheros”, “Subir TFG”, “Descargar normativa”, “Ver detalle TFG” y “listado”. Como muestra el gráfico, esta funcionalidad sólo esta disponible una vez que entras al detalle del TFG, que sera el nexo a todas las operaciones que puede realizar el estudiante.

3.2.2. Profesor

Toda la funcionalidad del usuario profesor viene recogida en este caso de uso que podemos ver en la imagen figura 3.2. El profesor, al igual que el estudiante, tiene unos casos de uso acordes con las acciones que puede realizar. Entre los requisitos asociados al profesor se encuentran: “Admitir defensa TFG”, “Generar informe TFG”, “Puntuar TFG”, “Seleccionar TFG”, “Crear Propuesta TFG” y “listado TFG”.

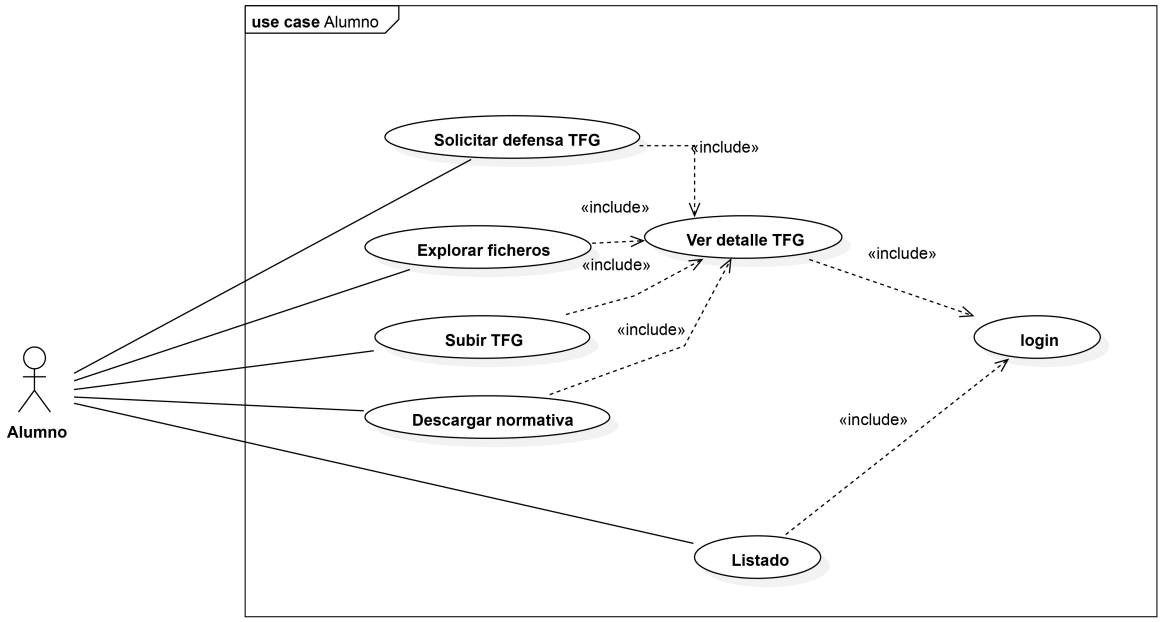


Figura 3.1: Caso de uso relacionado con la funcionalidad completa del alumno. Se puede comprobar que tiene acceso a la herramientas necesarias para gestionar su TFG.

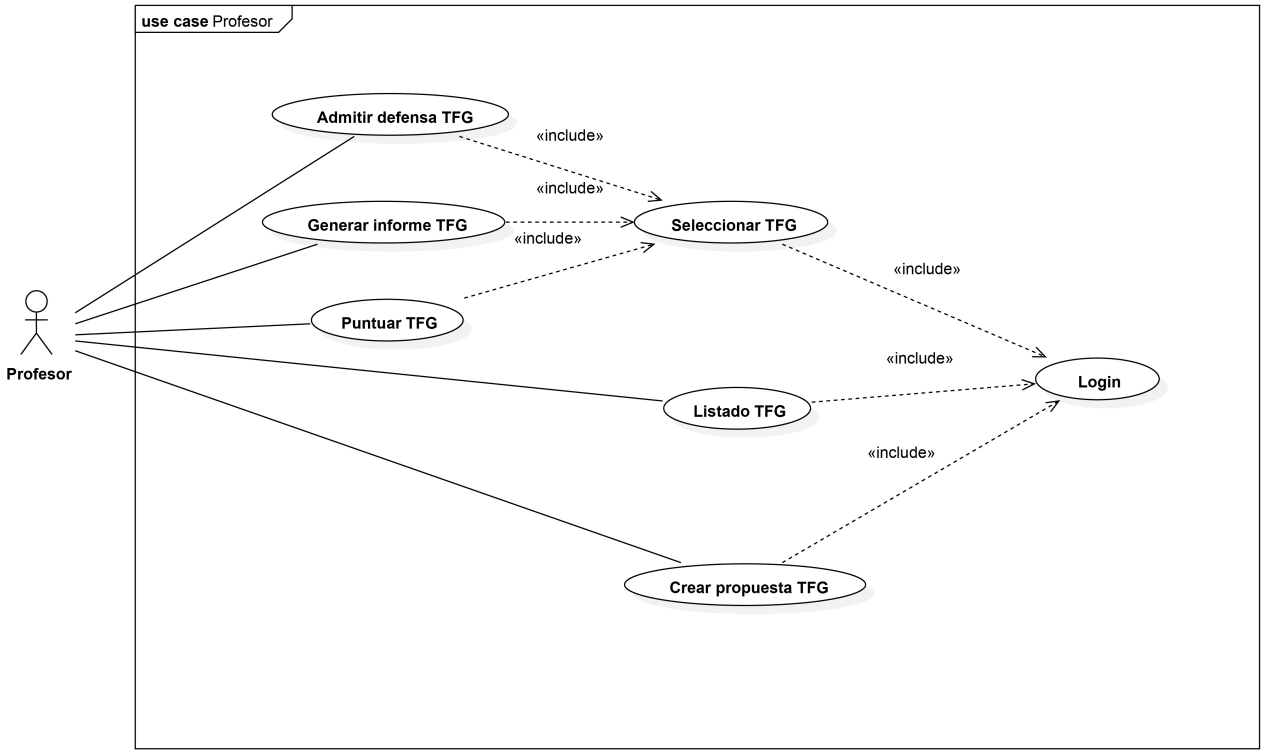


Figura 3.2: Caso de uso relacionado con la funcionalidad completa del profesor. Se puede observar las distintas funciones que puede realizar y que le permitirán generar el informe.

3.2.3. Tribunal

Este es el caso de uso relacionado con el tribunal. En este caso en la figura 3.3, se puede observar como el usuario tribunal dispone de bastante funcionalidad asociada y es que es uno de los actores principales en el tramite completo de un TFG. Entre las funcionalidades desarrolladas para la aplicación, el tribunal tiene los siguientes casos asociados: “Escribir anotacion TFG”, “Ver anotacion TFG”, “Descargar TFG”, “Puntuar TFG”, “Evaluar TFG”, “Calificar TFG”, “Admitir TFG”, “Seleccionar TFG”, “Listar TFG tribunal” “Listar TFG evaluados” y “Listar TFG”.

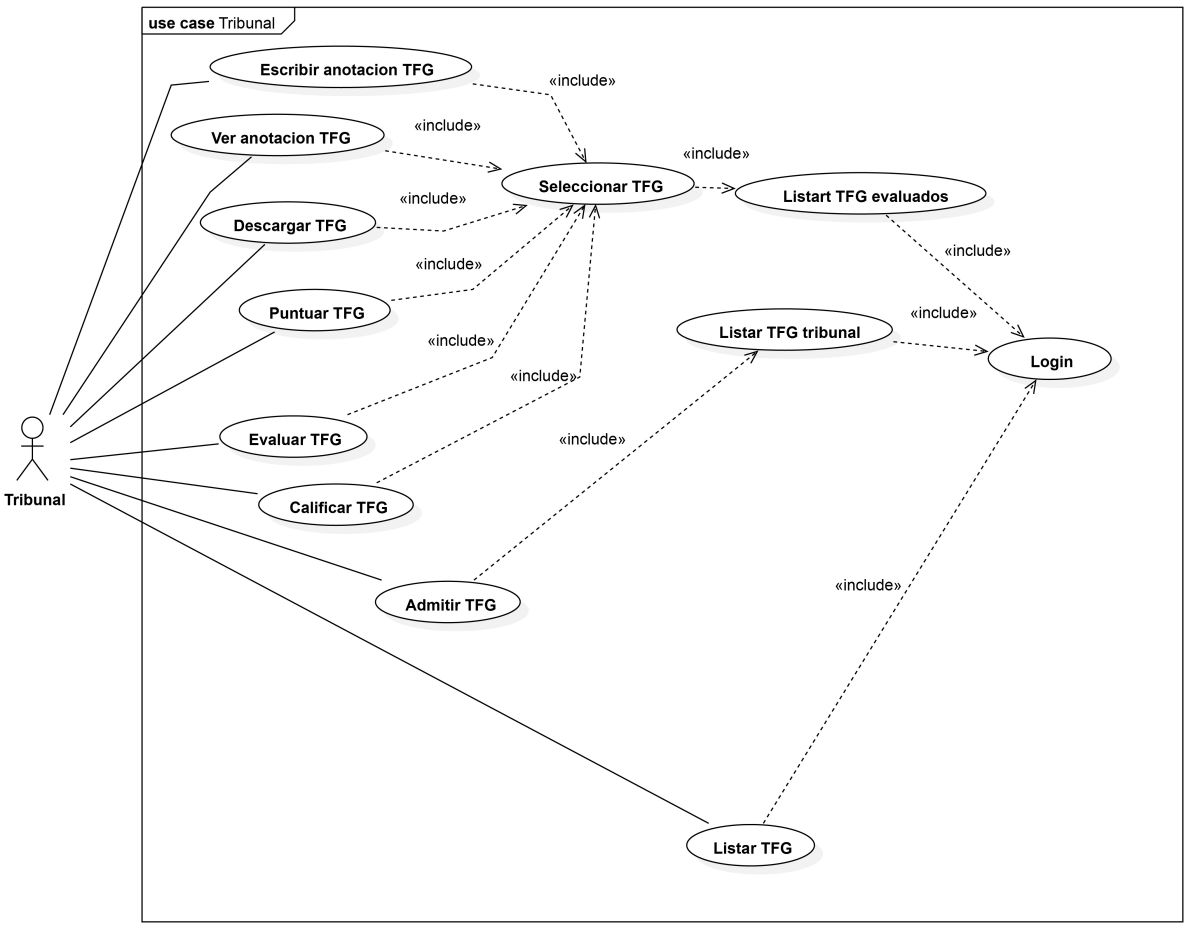


Figura 3.3: Caso de uso relacionado con la funcionalidad completa del tribunal. Este caso de uso es bastante extenso, ya que es un usuario que participa en varios estados, y su intervención sera crucial para que el TFG sea evaluado.

3.2.4. Administración

En este apartado se muestran los casos de uso relacionado con el usuario administración. Como se puede ver en la figura 3.4, este usuario dispone de una menor funcionalidad, pues solo tiene dos requisitos asociados, que son: “Admitir TFG” y “Hacer publico TFG”.

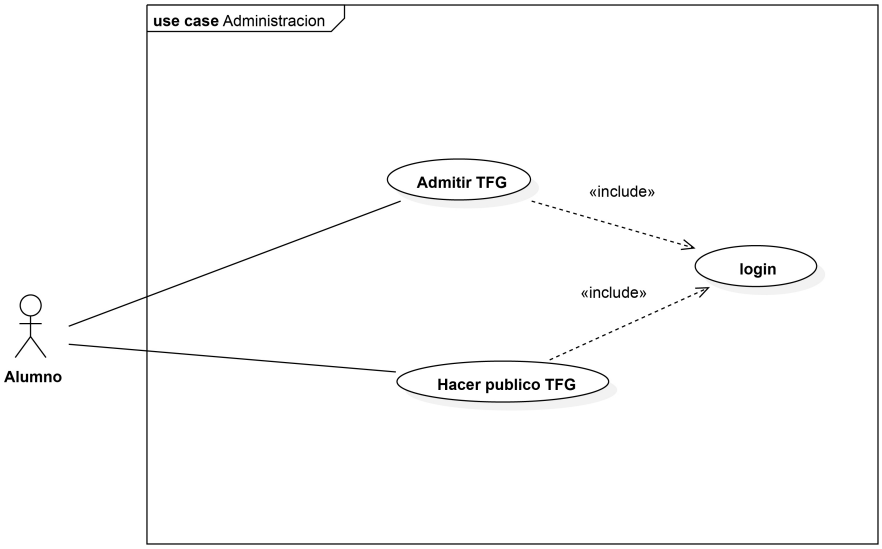


Figura 3.4: Caso de uso relacionado con la funcionalidad completa de la administración. Este usuario, aunque dispone de pocos casos, es crucial su competencia, pues permitira la admision del TFG.

3.2.5. Casos de uso detallado

En este apartado vamos a detallar tres de los casos de uso mas representativos del sistema.

CU01: Almacenar TFG

Nombre Caso de Uso: Almacenar TFG.

Actores primarios: Usuario “Estudiante”, registrado y conectado en la aplicación.

Interesados y objetivos:

- **Estudiante**: Usuario interesado en almacenar el TFG en el sistema para su correspondiente evaluación.

Resumen: El usuario conectado en la aplicación, y con un TFG asignado, accederá al detalle de su TFG y una vez abierto, podrá seleccionar el documento que desea almacenar en el sistema y pulsará en aceptar para guardar los cambios y así permitir el cambio del estado del TFG.

Precondiciones: El usuario estará registrado y conectado.

Garantía de éxito: El TFG se almacena en el sistema interno, y la base de datos se actualiza con

la información correspondiente.

Escenario de éxito:

1.– El usuario se conecta al sistema.
2.– El usuario selecciona su TFG.
3.– El usuario accede a su sistema para seleccionar el documento.
4.– El usuario selecciona el documento y lo acepta.
5.– El usuario confirma el documento seleccionado “Solicitud”.

Extensiones:

1.– El usuario se descarga la normativa.
2.– El usuario confirma la solicitud, pero esta no contiene el documento.
    2.1.– El sistema muestra un mensaje de error, y sugiere al usuario que seleccione un documento que cargar al sistema.

CU02: Calificar TFG

Nombre Caso de Uso: Calficar TFG.

Actores primarios: Usuario “Tribunal”, registrado y conectado en la aplicación.

Interesados y objetivos:

- **Tribunal**: Interesado en calificar el TFG para terminar con la evaluación.

Resumen: El usuario “Tribunal” tras esperar y obtener la evaluación de todos los integrantes del tribunal, procederá a calificar el TFG mediante un botón que le habilitara esa función.

Precondiciones: El usuario estará registrado y conectado.

Garantía de éxito: El TFG cambiara de estado a “Calificado”. Esa calificación se registrara en el sistema.

Escenario de éxito:

1.– El usuario se conecta al sistema.
2.– El usuario selecciona la lista de “Evaluados”.
3.– El usuario selecciona el TFG que quiere calificar.
4.– El usuario selecciona el boton que se le ha proporcionado, que califica el TFG.

Extensiones:

CU03: Admitir TFG

Nombre Caso de Uso: Admitir TFG.

Actores primarios: Usuario “Profesor”, registrado y conectado en la aplicación.

Interesados y objetivos:

- **Profesor**: Interesado en generar el informe sobre el TFG que tiene asignado.

Resumen: El usuario “Profesor” una vez que haya recibido la petición del estudiante para solicitar la defensa, deberá examinar el documento y en función de ello calificar y plasmara unas anotaciones. Tras ello deberá seleccionar el botón que genere el informe del TFG.

Precondiciones: El usuario estará registrado y conectado.

Garantía de éxito: El TFG cambia de estado permitiendo el avance de los tramites para evaluar el TFG.

Escenario de éxito:

1.– El usuario se conecta al sistema.
2.– El usuario selecciona el TFG sobre el que quiere generar el informe.
3.– El usuario plasmará anotaciones referentes al TFG.
4.– El usuario calificara de forma numérica el TFG.
5.– El usuario seleccionara el botón que generará el TFG.

Extensiones:

1.– El usuario genera el informe sin anotaciones ni calificación.
    1.1.– El sistema muestra un mensaje de error y no permite el avance de la aplicación.
2.– El usuario introduce caracteres alfabéticos.
    2.1.– El sistema muestra un mensaje de error indicando que solo aceptar caracteres numéricos.
3.– El usuario introduce caracteres numéricos con mas de dos cifras decimales.
    3.1.– El sistema muestra un mensaje de error indicando que solo aceptar dos caracteres numéricos.

3.3. Diagrama de flujo

3.3.1. Solicitud TFG Estudiante

En este diagrama (figura 3.5) podemos observar el comportamiento del sistema cuando un estudiante quiere solicitar la defensa de su TFG. Para ello el estudiante tiene que hacer login y seleccionar el TFG que tiene asignado. Una vez que entramos al detalle del TFG el estudiante podrá descargase la normativa en caso de que quiera revisar las condiciones antes de subir el TFG. Para subir el TFG debe examinar el fichero que tiene en su sistema y cargarlo a la aplicación. Una vez que esta cargado y el estudiante esté seguro de querer subir el TFG, solo debe seleccionar el botón “Solicitud”.

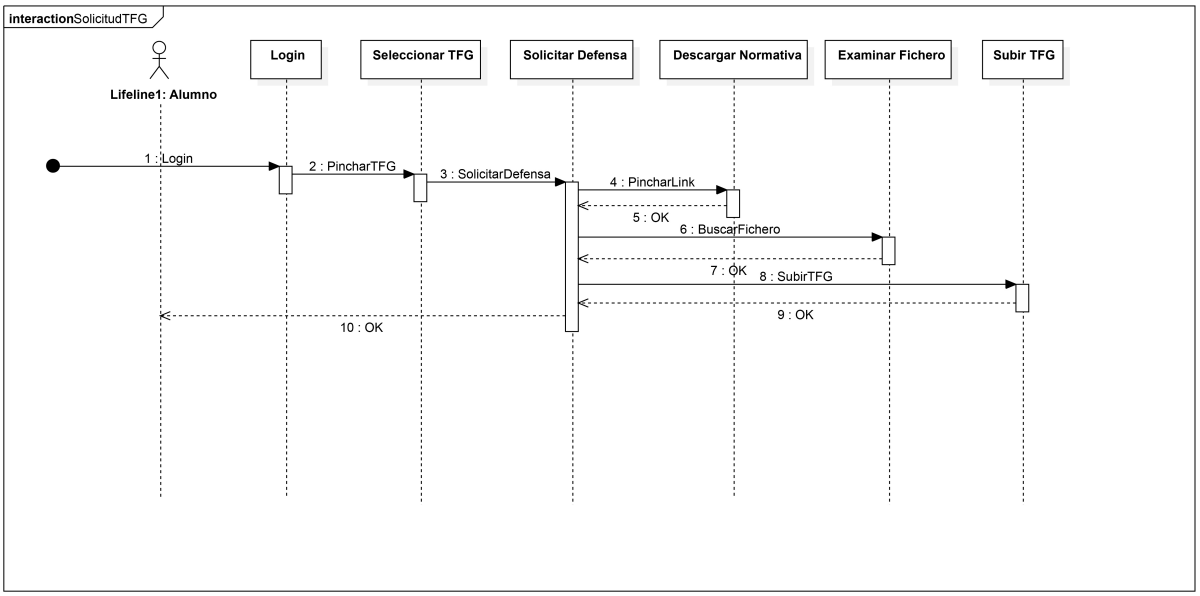


Figura 3.5: Imagen asociada a un diagrama donde podemos ver el proceso completo de la solicitud de un TFG por parte de un estudiante. En el se puede observar el intercambio de mensajes que ocurre en el sistema y la respuesta de este.

La comunicación entre los distintos módulos es interna, intercambiando de este modo objetos que serán necesarios para el correcto funcionamiento de la aplicación. En este caso solo habrá dos acciones que creará una petición “http” generando de “OK” cuando la petición se resuelva de forma correcta.

3.3.2. Admitir TFG Profesor

Tras completar el estado anterior, el profesor deberá validar el TFG y aceptar la tramitación del TFG. Para ello y tal como se puede observar en la figura 3.6 después de hacer login, el usuario debe seleccionar el TFG que esta listo para ser defendido. Una vez accedido el profesor debe rellenar un campo donde plasmará las anotaciones sobre la memoria y al mismo tiempo reflejará ya la nota que asigna sobre ese trabajo. Cuando este todo asignado, podrá seleccionar botón continuar para que el TFG cambie de estado. En este caso, el numero de módulos implicados es menor que con respecto al estudiante, pero el tipo de comunicación y respuesta es el mismo. De este modo el sistema responderá con “OK” cuando termine de gestionarse la generación del informe.

3.3.3. Seleccionar Tribunal

Esta es una funcionalidad única de la administración. En la figura 3.7 se muestra que serán los encargados de generar el comité de profesores que pasaran a formar del tribunal. Para ello, debe autenticarse en la aplicación y seleccionar la creación de tribunal. Cuando el usuario esté en la nueva

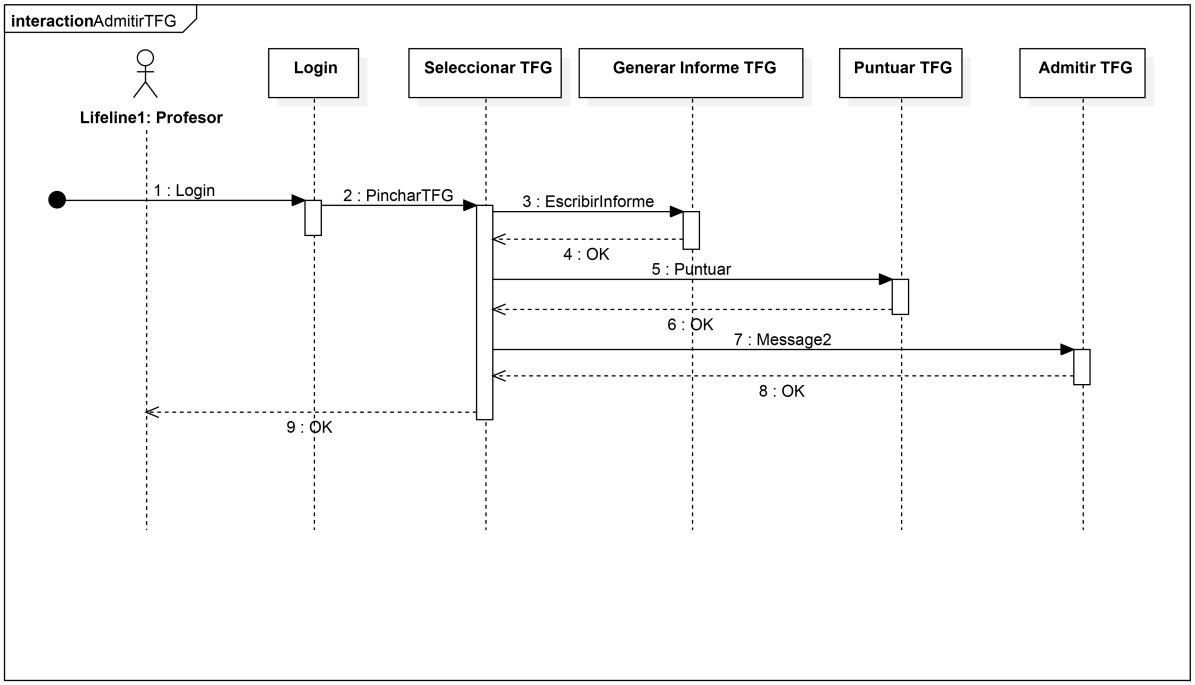


Figura 3.6: Diagrama correspondiente con el flujo seguido en la aplicación para que un profesor admita el TFG. Se puede observar el intercambio de mensajes necesario entre los distintos modulos para el correcto funcionamiento del sistema.

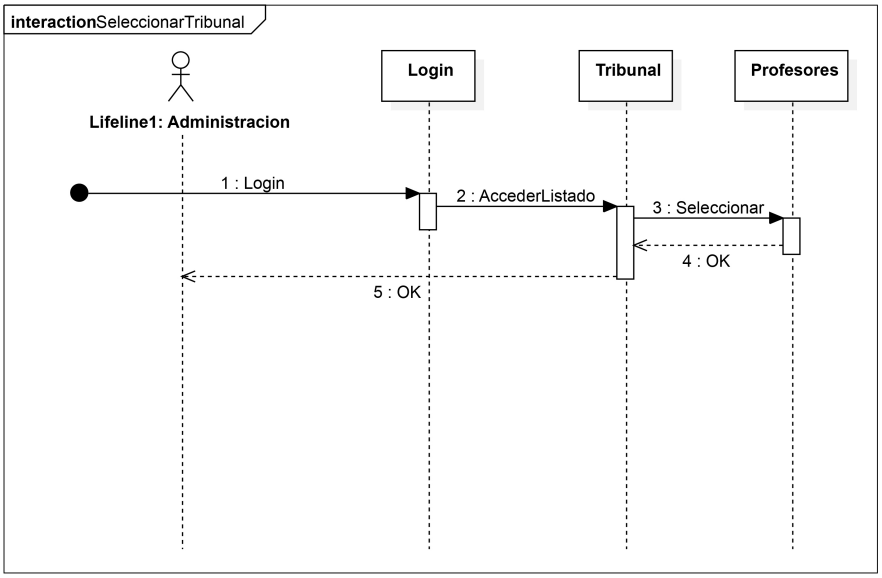


Figura 3.7: Diagrama asociado a la seleccion de los integrante de los tribunales que se puedan formar. El intercambio de mensajes es breve y el sistema responderá

ventana, verán un listado con los profesores de la facultad y deberán elegir a un total de 12 que pasan a formar parte del tribunal. Aunque el flujo no sea muy extenso es de extrema relevancia, ya que la selección de los profesores que estén habilitados para formar un tribunal quedará almacenada en la base de datos.

3.3.4. Admitir TFG Administración

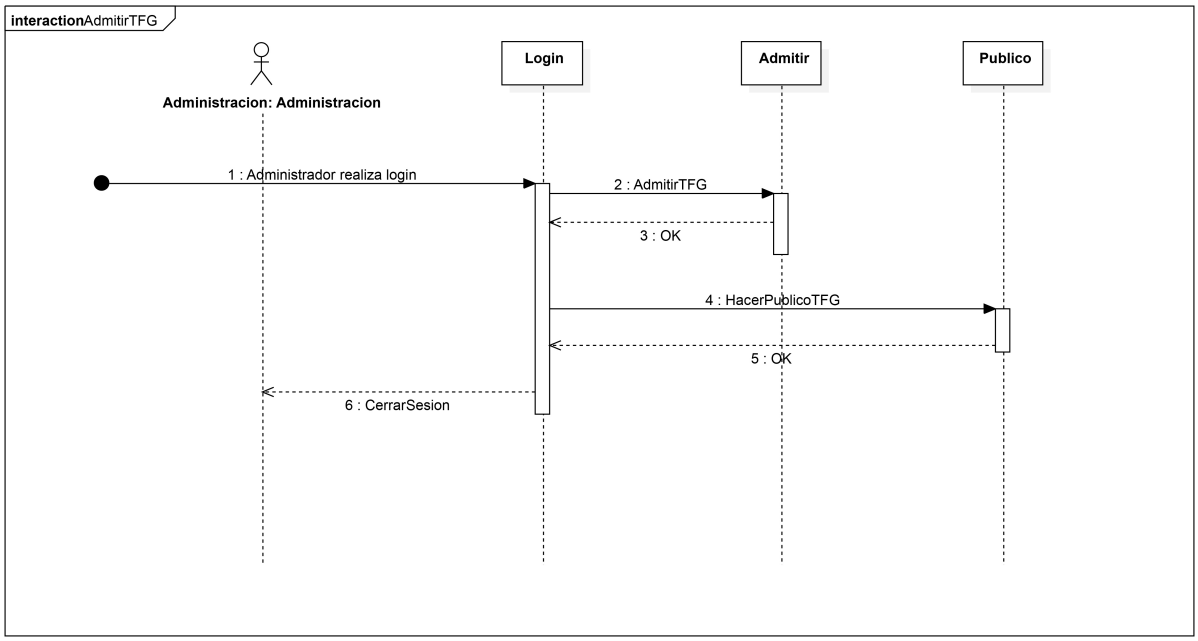


Figura 3.8: Diagrama asociado a la acción de admitir un TFG por parte de la administración. Breve intercambio de mensajes que muestra el flujo para la admisión del TFG por parte de la administracion.

Como se puede ver en la imagen figura 3.8 administración tiene un usuario que sera uno de los encargados de admitir el TFG para ser evaluado. Además tiene la capacidad de hacer público ese TFG. Tras este breve intercambio de mensajes, el sistema responde al usuario informando sobre el estado de su petición. Mientras internamente la información se almacena mediante un registro en la base de datos.

3.3.5. Admitir TFG Tribunal

Tras aceptar el profesor el TFG para ser defendido, el tribunal deberá admitirlo para la evaluación. Para ello deberá loguearse en la aplicación y acceder al listado de TFG que están a la espera de ser admitidos. El presidente del tribunal deberá seleccionar el TFG que quiere admitir para evaluación y aceptar. Se puede ver reflejado en la imagen figura 3.9.

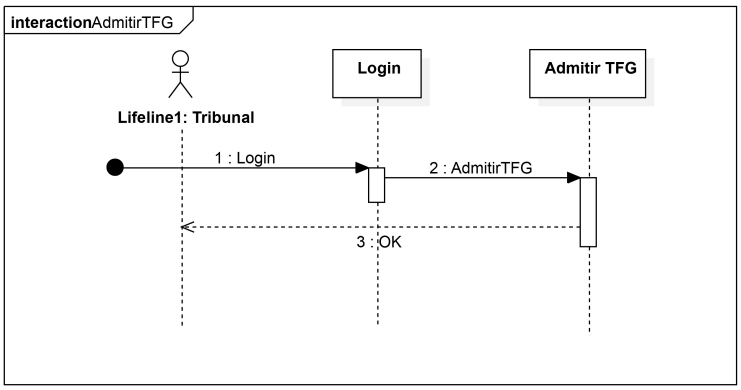


Figura 3.9: Diagrama asociado a la admisión del TFG por parte del presidente del tribunal. El solo debe seleccionar el TFG que quiere admitir y el sistema respondera con el resultado de la operación.

3.3.6. Evaluar TFG Tribunal

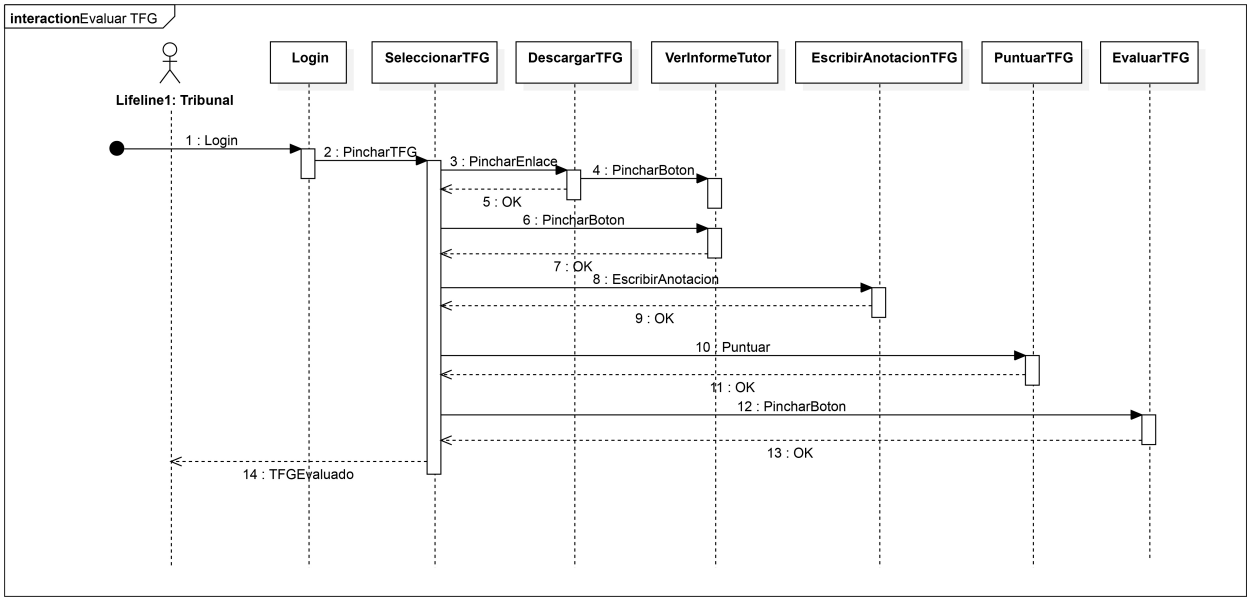


Figura 3.10: Diagrama asociado a la evaluación de un TFG por parte del tribunal. Como se puede observar, el intercambio de información entre los distintos módulos es extensa, pues tanto “DescargarTFG” como “VerInformeTutor” y “EvaluarTFG” generan respuestas “http” que deben ser tratadas.

Una vez que el TFG ha sido admitido para evaluación, el tribunal sera el encargado de asignar la nota. Para ello deberán loguearse en el sistema y acceder al listado de los TFG que están listos para evaluar. Una vez en el listado, el tribunal deberá acceder al TFG que quiere evaluar. Dentro del detalle, el tribunal podrá descargar el TFG del estudiante y la anotación escrita por el profesor. Una vez terminada la defensa, podrán escribir una anotación y plasmar la nota que deseen. Cuando los 3 integrantes del tribunal hayan escrito su nota, el presidente deberá acceder al TFG y presionar sobre el botón “calificar”. El TFG estará calificado como se ve en la figura 3.10.

Al igual que en los demás flujos, en este el intercambio de objetos entre los módulos es constante, y son necesarios para construir la lógica de la aplicación. Además el sistema contestará mediante peticiones “http” al usuario que informará del estado de la gestión.

3.4. Diseño Base de Datos

La base de datos, como ya se detallo anteriormente, es una base de datos relacional sostenida por **SQLite**. La particularidad de este tipo de base de datos es que sigue el **modelo relacional** [14]. Este modelo se basa en la organización y gestión de los datos en “tablas” y se fundamenta en el uso de relaciones o conjunto de datos, lo que hace que sean conocidas normalmente por el nombre de tablas.

Estas tablas (relaciones) están formadas por tuplas, que hacen la vez de filas, y a su vez por campos, que corresponden a las columnas de la tabla. La información que se almacena en estas tuplas son los registros. Cada tabla por definición debe poseer un clave primaria, que es un identificador único de esa relación. Además puede contener claves externas, que marca la relación con otra tabla.

Para esta aplicación se ha seguido usando el sistema de base de datos relacional que ya estaba implementado dado que la base de datos es incremental, lo datos deben ser consistentes y la relación entre las entidades es estrecha. Para mas detalle, en la figura 3.11 se puede observar como están determinadas estas relaciones.

Si se observan las tablas, podemos ver como la tabla, con mayor información de todas y sobre la que actúa la mayor parte de la lógica, es “tfg”. Esta tabla contiene como primary key el código del tfg, que es único e irrepetible, y como claves foráneas las correspondientes al tutor y al estudiante. Por otro lado, la tabla tribunal contiene toda la información referente a la lógica del tribunal, como pueden ser los campos anotaciones y nota, que serán fundamentales para calificar un TFG. La clave foránea de esta tabla es “codigo” de la tabla TFG.

En la tabla usuario se almacena la información de todos los usuarios del sistema, tanto profesores como estudiantes, salvo el usuario “administración”. En esta tabla, la clave primaria es el ID.

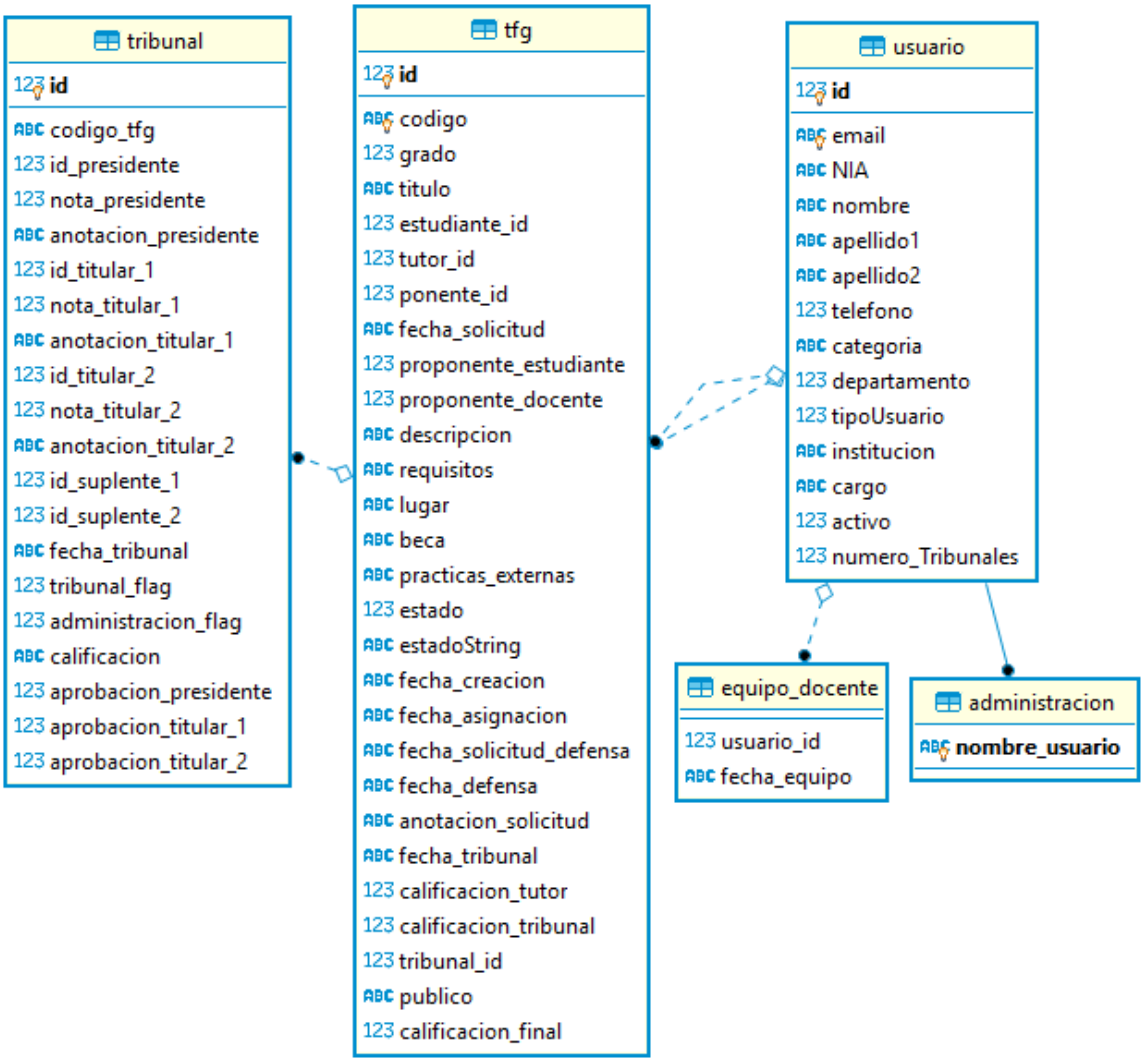


Figura 3.11: Esquema relacional de la base de datos usada en la aplicación

4

Desarrollo

La metodologia de trabajo seguida es importante, pues sera parte fundamental a la hora de desarrollar el trabajo y depende de la metodologia escojida, el desarrollo sera mas eficiente.

Como se comentó al principio del TFG, la metodologia del desarrollo del TFG ha seguido los criterios de la metodologia Agile [15]. Esta metodología, en este proyecto, ha facilitado el desarrollo de la aplicación, ya que las reuniones que se han matenido durante la creacion han permitido avanzar sin entorpecer con las obligaciones que teniamos ninguno de los implicados.

4.1. Metodología Agile aplicada al proyecto

Siguiendo con las bases de la **metodología Agile**, el equipo de trabajo que forma parte de el es el siguiente:

4.2. Equipo de trabajo

La comunicación y la labor de equipo es fundamental en este tipo de metodologías, así como la asignación de tareas acordes con el rol y la organización de estas para lograr los objetivos marcados. En este tipo de metodología, podemos encontrar 3 roles distintos.

4.2.1. Product Owner

Esta aplicacion como se ha detallado, parte de un estado en el cual ya estaba en funcionamiento. Durante ese periodo la persona que ha hecho las veces de **Product Owner** es Oscar Delgado.

El Product Owner es el rol encargado de mantener contacto con el cliente y garantiza que el resultado del desarrollo esta acorde de las exigencias del cliente. En esta aplicación Óscar es el dueño del producto y es la persona sobre la que recae las decisiones finales del desarrollo.

4.2.2. Scrum Master

Por otro lado, este tipo de metodología contiene otro rol, el **Scrum Master**. Este rol se encarga de que la metodología sea entendida por el equipo de trabajo y asegura que los objetivos implantados en el sprint se alcancen.

En esta aplicación, ese rol ha sido asignado para Eloy Anguiano debido a su conocimiento con respecto a las tecnologías empleadas en esta aplicación y por el papel de tutor del TFG cuyas tareas se acoplan a la perfección con este rol.

4.2.3. Equipo de Desarrollo

En una metodologia scrum, el equipo de desarrollo es el encargado de implementar los objetivos marcados por el Product Owner. Este rol es el que desarrolla e implementa la solución y funciona de manera autónoma con las directrices marcadas.

En este caso el papel del equipo de desarrollo esta asignado a Óscar Ruiz ya que es el actor que esta desarrollando e implementando la aplicación.

4.3. Sprints

La metodología empleada cuenta con una serie de reuniones esenciales para asegurar el correcto desarrollo y avance de la aplicación. Siguiendo con la metodología detallada, en cada reunión *sprint* se iban detallando las nuevas funcionalidades que estaban relacionadas con los estados por los que un TFG avanza.

4.3.1. Solicitud

A continuacion se vas a exponer los requisitos funcionales desarrollados en el primer sprint, que son los siguientes:

Requisitos funcionales implicados en este sprint

RF-4, RF-5, RF-6, RF-7, RF-8

Esta fue la primera *release* que daba inicio al TFG. Para esta el objetivo era aprender y adaptarse a la plataforma y el programa. Los objetivos marcados eran conseguir que el estudiante almacenara el TFG en el sistema y aumentar el numero de estados del TFG.

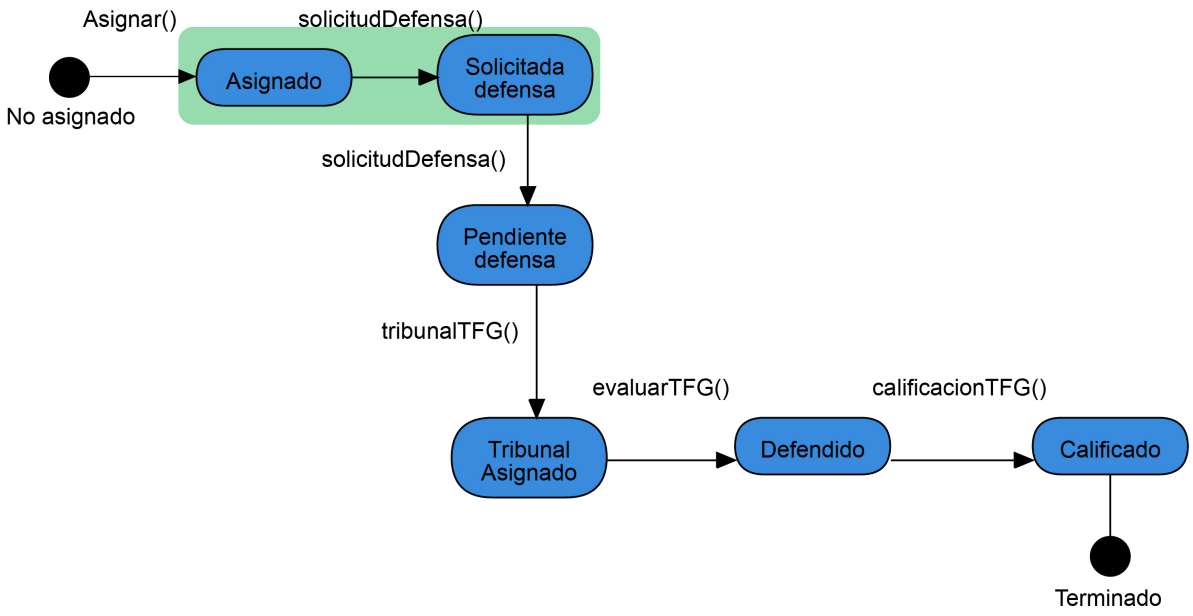


Figura 4.1: Diagrama de estado que muestra la transición entre el estado de Asignado a Defensa solicitada. El estado cambia una vez que el estudiante cumplimenta los campos para solicitar la defensa.

El estado que se habilitó en esta release es el de "solicitado". Cuando un estudiante quiere optar a la defensa de su trabajo, es necesario que suba el TFG una vez que lo ha acordado con su tutor. De esta forma el trabajo se almacena en el sistema y ya esta accesible para que sea leído y examinado por el tutor. Como se puede observar en la figura 4.1 el diagrama representa en esta fase el cambio de estado de “Asignado” a “Solicitada defensa”. Esto ocurrirá después de que el estudiante termine el proceso de subir su TFG.

De forma interna el sistema recoge la petición del estudiante y la procesa para almacenar los datos referentes con la petición del estudiante, como es en este caso la fecha de petición y el cambio de estado. Por otro lado el documento de la memoria se almacena en un repositorio interno de TFG usando el nombre del estudiante y el código del TFG asegurando de esta forma la correcta asignación de la memoria con el estudiante. Una muestra de como se trata la información lo podemos ver en el apartado 4.3.1.

Una vez que el TFG se encuentra en el sistema, el estado del TFG se actualiza como hemos indicado en la figura 4.1. El siguiente estado al que debe avanzar el TFG es al de “Pendiente Defensa” como se muestra en la figura 4.2. Tras el avance del estado “Solicitada defensa”, el profesor se encuentra con el sistema listo para que pueda plasmar las anotaciones sobre el TFG y además registrar la calificación asociada a ese TFG como se muestra en el apartado 4.3.1.

Código 4.1: Esta figura contiene el código correspondiente a la solicitud de defensa por parte del estudiante. Cobra especial importancia la forma de nombrar al fichero y como se almacena la información en la base de datos.

```
1        tfg, tutor, ponente, estudiante = self.__getDatosTFG(tfg_codigo)
2        user = cherrypy.session['user']
3        if 'ufile' in kwargs:
4
5            upload_path = os.path.dirname(__file__)
6            absDir = os.path.join(os.getcwd(), upload_path)
7            user = cherrypy.session['user']
8
9            if cherrypy.request.method == 'POST' and\
10               'ufile' in kwargs:
11               upload_file = os.path.join(upload_path, 'TFG_' + user.nombre + '_' + user.apellido1 +
                         '_' + user.apellido2 + '_' + tfg_codigo)
12
13               size = 0
14               with open(upload_file, 'wb') as out:
15                   while True:
16                       data = kwargs['ufile'].file.read(8192)
17                       if not data:
18                           break
19
20                       out.write(data)
21                       size += len(data)
22               out = ''
23
24               tfg.setEstado(TFG.SOLICITADA_DEFENSA)
25               tfg.fecha_solicitud_defensa = datetime.date.today().strftime("%Y/%m/%d")
26               self.db.commit()
27               raise cherrypy.HTTPRedirect("home")
28
29
30           resultado = self.db.query(TFG, Usuario).\
31                           filter(TFG.tutor_id==Usuario.id).\
32                           filter(TFG.estado == TFG.NO_ASIGNADO).all()
```

Código 4.2: En esta figura podemos encontrar el código que valida y cambia el estado del TFG para ser admitido por la administración y el tribunal.

```
1    tfg.anotacion_solicitud = kwargs['anotacion_solicitud']
2    if user.email.endswith('@uam.es'):
3
4        tfg.calificacion_tutor = kwargs['nota']
5
6    tfg.setEstado(TFG.PENDIENTE_DEFENSA)
7    tfg.fecha_defensa = datetime.date.today().strftime("%Y/%m/%d")
8        # Guardamos los cambios
9    self.db.add(tfg)
10   self.db.commit()
```

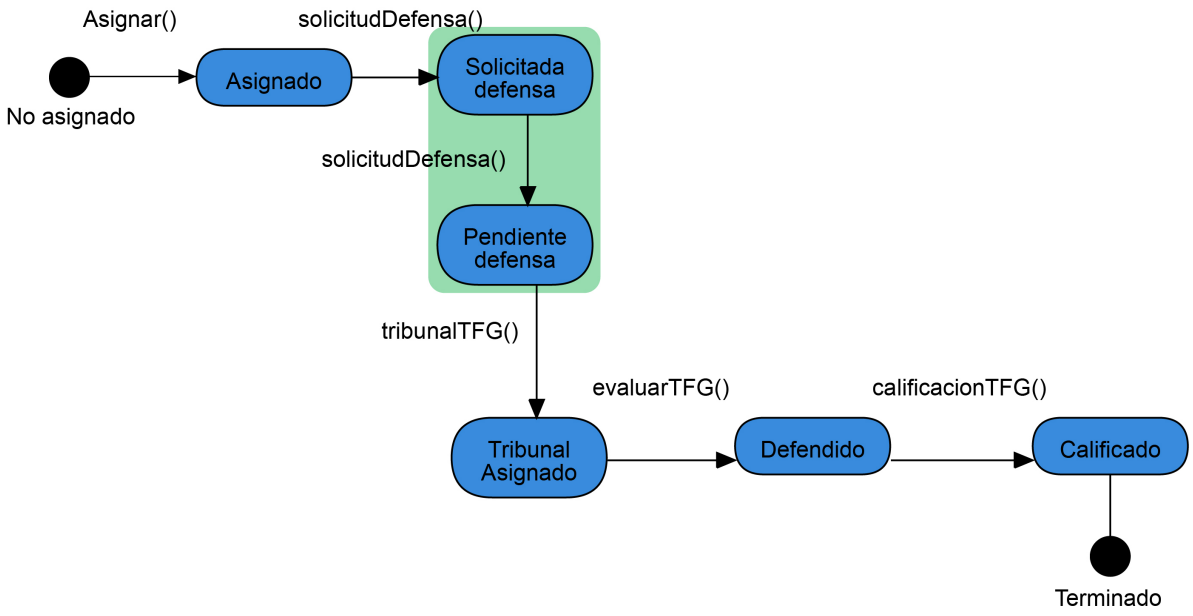


Figura 4.2: Diagrama de estado que muestra la transición entre el estado de Defensa solicitada a Pendiente defensa. Cambio mostrado en el código correspondiente.

4.3.2. Admitido

Los requisitos funcionales desarrollados en el segundo *sprint* son los siguientes:

Requisitos funcionales implicados en este *sprint*

RF-10, RF-11, RF-12

El siguiente estado al que debe avanzar el sistema es al de “Tribunal asignado”. Esta *release* tiene un componente añadido con respecto a las otras, y es que en este caso son dos los usuarios los implicados. Ambos deberán admitir el TFG para que pueda cambiar de estado.figura 4.2.

En esta fase de la aplicación, se necesita que el administrador admita a tramite el TFG para que este pueda seguir avanzando con el flujo normal, además tiene la funcionalidad añadida que le permite hacer publico o no un TFG después de la defensa. Ambas validaciones se almacenan como una como un registro en la base de datos, que recogerá las respuestas de la petición realizada por el administrador como se puede ver en el apartado 4.3.2. Tras admitir el TFG, el usuario administración debe seleccionar, entre una lista de docentes disponibles, quienes serán los integrantes del tribunal. Este selección es irreversible ya que tras aceptar, se crea una nueva tupla en la tabla “Tribunal” donde se almacena los integrantes del tribunal y la fecha de creación de este como se ve en el apartado 4.3.2.

Una vez asignado el tribunal, el presidente, previamente asignado por la administración, podrá

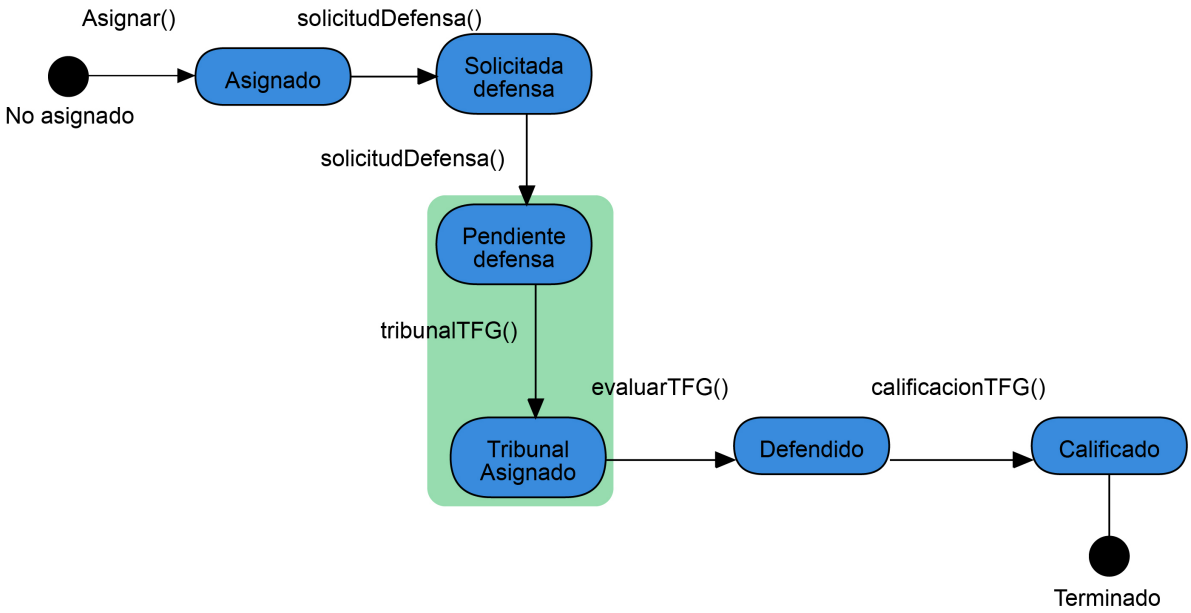


Figura 4.3: Diagrama de estado que muestra la transición entre el estado de Pendiente defensa a Tribunal asignado. La transición se hará efectiva cuando la administración y el presidente del tribunal acepten la admisión.

Código 4.3: En esta figura se presenta el código correspondiente a la admision del TFG por parte del tribunal.

```
1  if 'btn_aceptar' and 'publico' in kwargs:
2
3      seleccionados = kwargs["publico"]
4      if type(seleccionados) is not list:
5          tfg = self.db.query(TFG).filter(TFG.codigo == seleccionados).first()
6          tfg.setPublico()
7          self.db.commit()
8      else:
9          for publico in seleccionados:
10             tfg = self.db.query(TFG).filter(TFG.codigo == publico).first()
11             tfg.setPublico()
12             self.db.commit()
13
14 if 'btn_aceptar' and 'codigo' in kwargs:
15
16     seleccionados = kwargs['codigo']
17     if isinstance(seleccionados, list):
18         raise cherrypy.HTTPRedirect("asignacionTribunalTFG?seleccionados="+'/'.join(seleccionados))
19     else:
20         raise cherrypy.HTTPRedirect("asignacionTribunalTFG?seleccionados="+seleccionados)
```

Código 4.4: En esta figura se presenta el código correspondiente a la creación y almacenamiento del tribunal en la base de datos.

```
list_presidente = [tribunal[0]]
list_titulares = [tribunal[1], tribunal[2]]
list_suplentes = [tribunal[3], tribunal[4]]
new_tribunal = Tribunal(codigo_tfg = codigo, id_presidente=tribunal[0], id_titular_1 = tribunal[1],
            id_titular_2 = tribunal[2], id_suplente_1 = tribunal[3], id_suplente_2 = tribunal[4],
            fecha_tribunal = datetime.date.today().strftime("%Y/%m/%d"),
            administracion_flag = 1)

tfg.fecha_tribunal = datetime.date.today().strftime("%Y/%m/%d")
self.db.add(new_tribunal)
self.db.commit()
raise cherrypy.HTTPRedirect("home")
```

admitir el TFG. Esto significa el cambio de estado del TFG a “Tribunal asignado”. La admisión del TFG por parte del tribunal queda recogido del mismo modo que ocurría con el usuario administrador mediante un registro en la base de datos en la tabla “Tribunal”.

4.3.3. Evaluado

Los requisitos funcionales desarrollados en el tercer sprint son los siguientes:

Requisitos funcionales implicados en este sprint

RF-13, RF-14, RF-15

Una vez que el tribunal ha sido formado el estado del TFG pasa a ser el de “Tribunal asignado”. En este sprint la tarea abordada fue la de generar un nuevo estado, “Defendido” figura 4.4.

En el estado actual, todos los profesores que forman el tribunal pueden acceder a el a través del detalle del mismo. En este punto los integrantes tienen habilitado los campos necesarios para escribir las anotaciones correspondiente con la defensa y asignar una nota que evaluara el TFG. Para ello la base de datos esta preparada para almacenar tantos las anotaciones como la nota. Una vez que los 3 integrantes del TFG han plasmado su veredicto, el estado del TFG cambiara a “Defendido” como queda recogido en el apartado 4.3.3.

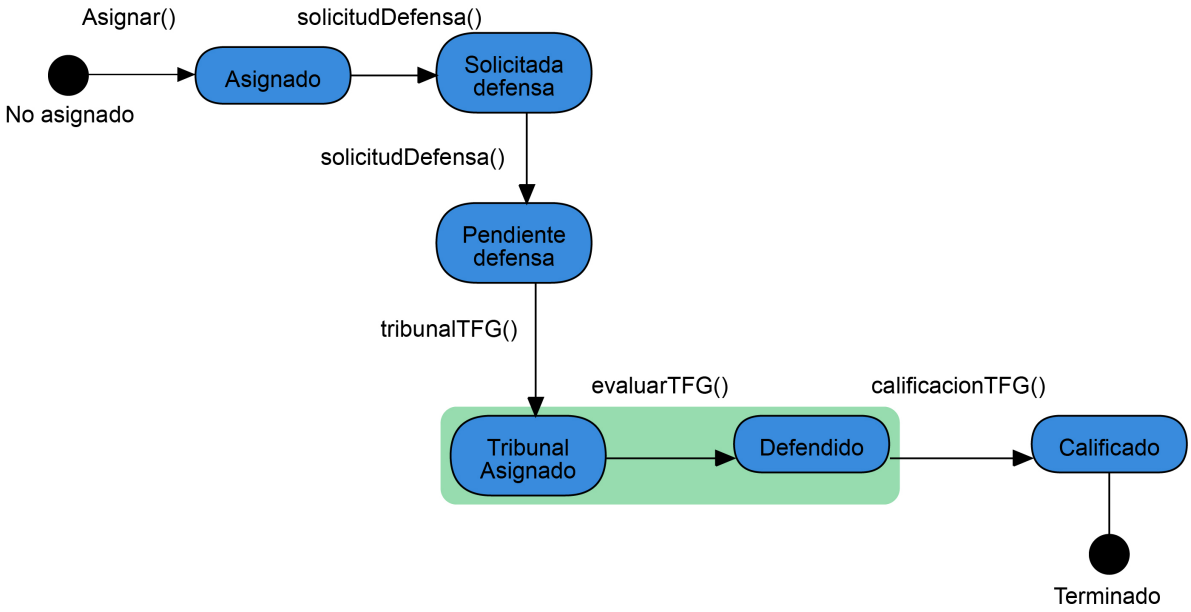


Figura 4.4: Diagrama de estado que muestra la transición entre el estado de Tribunal asignado a Defendido. Esta transición se hará efectiva una vez que los 3 integrantes del tribunal hayan evaluado el TFG.

Código 4.5: En la figura mostrada, se puede observar como en el código se modifica el estado del TFG cuando se posee la nota de los tres integrantes del tribunal.

```
1  tribunal = self.db.query(Tribunal).filter(Tribunal.codigo_tfg == tfg_codigo).first()
2
3  if tribunal.nota_presidente and tribunal.nota_titular_1 and tribunal.nota_titular_2:
4
5      tfg = self.db.query(TFG).filter(TFG.codigo == tfg_codigo).first()
6      tfg.setEstado(TFG.DEFENDIDO)
7
8  self.db.commit()
9
10 if cherrypy.request.method == 'POST':
11
12     raise cherrypy.HTTPRedirect("evaluacionTFG")
```

4.3.4. Calificado

Los requisitos funcionales desarrollados en el ultimo sprint son los siguientes:

Requisitos funcionales implicados en este sprint

RF-16

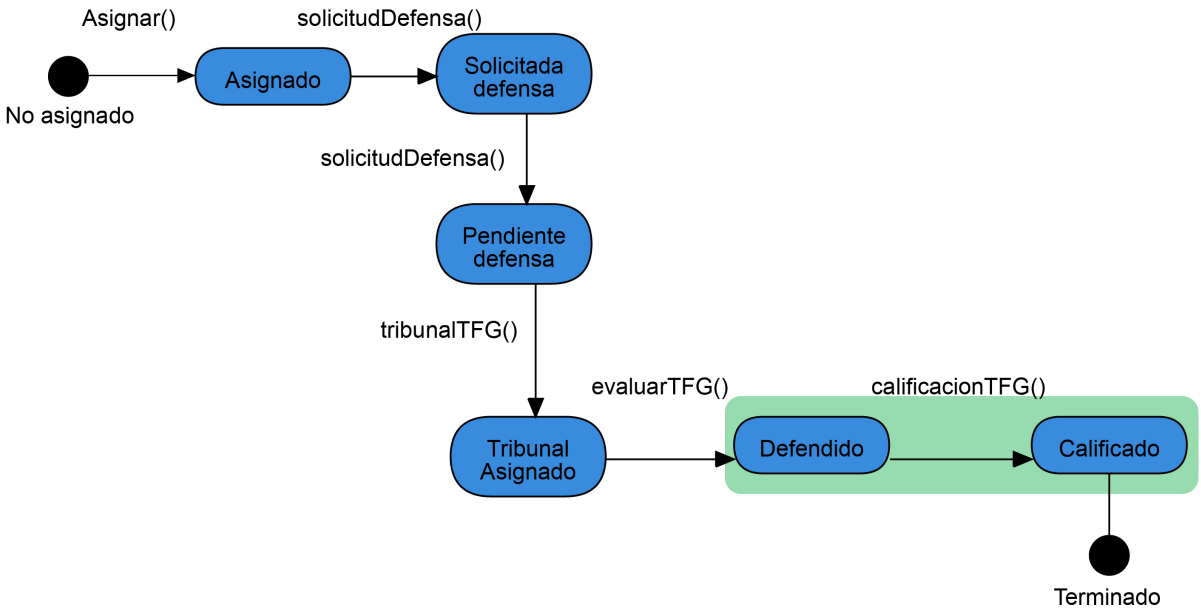


Figura 4.5: Diagrama de estado que muestra la transición entre el estado de Defendido a Calificado. Este es el último estado al que el sistema llega.

En este *sprint* se abordó el ultimo estado del sistema “Calificado” visto en la figura 4.5. Este es el estado final del sistema, en el cual el TFG quedará ya calificado y registrado en el sistema. Para ello solo el presidente del tribunal tiene la potestad para calificar el tribunal. Una vez que se califica, en la tabla TFG se registra la fecha y la nota total del TFG con las medias y la ponderación total de cada profesor. apartado 4.3.4

Código 4.6: En el código mostrado se puede ver cual es la lógica seguida para generar la calificación numerica. Una vez generada la calificación queda almacenada en la base de datos.

```
tribunal = self.db.query(Tribunal).filter(Tribunal.codigo_tfg == tfg_codigo).first()
user = cherrypy.session['user'];
nota = tribunal.nota_presidente + tribunal.nota_titular_1 + tribunal.nota_titular_2
nota = (nota / 3)

if tfg.calificacion_tutor != None:

    nota_tribunal = nota *0.6
    nota_tutor = tfg.calificacion_tutor *0.4
    final = nota_tutor + nota_tribunal
else:

    final = nota

final = str(round(final, 1))
tfg.setCalificacionTribunal(nota)
tfg.setCalificacionFinal(final)
tfg.setEstado(TFG.CALIFICADO)
self.db.commit()
raise cherrypy.HTTPRedirect("calificacionTFG?tfg_codigo="+tfg.codigo)
```

5 Integración, Pruebas y Resultados

Durante todo el desarrollo de la aplicación al mismo tiempo que se iba avanzando, se realizaban distintas pruebas que iban validando el resultado del desarrollo. La integración al principio resultó algo costosa al no ser capaz de hacer funcionar correctamente el entorno en un PC que contiene hardware obsoleto, pero tras instalar las librerías correctas no hubo mayor problema.

En este proceso de validación ha sido participe el cliente, que en este caso es el tutor del TFG y el propietario de la aplicación.

5.1. Pruebas con Cliente

Estas pruebas fueron realizadas de forma presencial con las personas implicadas en esta aplicación. En cada reunión de *sprint*, además de discutir los hitos a realizar en las siguientes iteraciones, se realizaba una serie de test sobre el desarrollo llevado a cabo en la iteración para validar el avance y marcar los errores a corregir.

5.2. Pruebas de Sistema

5.2.1. Solicitud

En este *sprint* además de cerrar los objetivos del siguiente desarrollo, se validó mediante un conjunto de pruebas en directo, el correcto funcionamiento de desarrollo del estado de “Solicitud defensa”. Para ello se comprobó de forma visual la navegabilidad del usuario dentro del sistema y de como el objetivo marcado, que era comprobar que un TFG cambiaba de estado una vez que el TFG se almacenaba a la plataforma se realizaba. Por otro lado y de forma interna, se demostró la actualización de la información en la base de datos y del almacenamiento del mismo trabajo dentro del sistema siguiendo la convicción establecida con el nombre del TFG.

5.2.2. Lectura

En este caso, la comprobación que se realizo fue la de verificar que el nuevo estado era funcional y que un usuario profesor podía acceder al detalle del TFG del cual era el tutor. Además una vez accedido al detalle, se debía comprobar que el usuario era capaz de realizar el informe del tutor y la valoración cuantitativa del trabajo. Desde el punto de vista del sistema, se debía comprobar si la base de datos se actualizaba con el nuevo estado y almacenaba los campos rellenados por el tutor.

5.2.3. Admitido

La validación de este estado consistió en la demostración visual del cambio de estado del TFG tras ser admitido por administración y el presidente del tribunal. Para esta evolución de la funcionalidad, se creó una nueva tabla en la base de datos que dejaba preparado el sistema para gestionar el tribunal asignado al TFG.

En este *sprint* se debía validar que el administrador fuera capaz de admitir a tribunal el TFG y de chequear la opción de que fuese publico. Una vez hecho esto, se comprobó que la transición de estado de admitido relacionada con el administrador y el campo “publico” que permite que un TFG se haga publico, se activara. Por otro lado, se comprobó que el administrador era capaz de crear el tribunal con los componentes necesarios para formarlo, asignándoles un rol distinto. Una vez creado en la aplicación, se comprobó que el tribunal quedo reflejad en la base de datos.

Por ultimo, el nuevo tribunal creado, se demostró que el rol de presidente del tribunal era capaz de admitir el TFG y que tras admitirlo, se registraba dentro de la tabla Tribunal el valor correspondiente que marca que el tribunal admite el TFG.

5.2.4. Tribunal

En este “sprint”, el desarrollo a evaluar era la interacción del tribunal con el TFG creado. Se comprobó, de forma visual, que cada integrante del tribunal podía seleccionar y acceder al detalle. Este sprint estuvo más centrado en la mejora y corrección del desarrollo de la aplicación. Además se mostró el diseño de la nueva tabla “Equipo docente” que almacena los profesores que están habilitados para formar parte del tribunal.

5.2.5. Evaluado

Este fue uno de los últimos *sprint*. donde se evaluó la capacidad de los integrantes del tribunal para poder asignar el informe del tribunal y la calificación al TFG del estudiante. Para ello se demostró que

cada integrante era capaz de acceder al detalle del TFG y que la pantalla mostrada era la de evaluación. Además se enseñó que los integrantes pueden descargarse el TFG y acceder a la anotación del tutor. Internamente la información que registraba los profesores es almacenada en la tabla “tribunal” de la base de datos junto con la fecha de la presentación.

5.2.6. Calificado

En el ultimo *sprint* se revisó el funcionamiento completo de la aplicación, pero en concreto se comprobó que el último estado funcionaba correctamente. En este caso se mostró como el TFG cambiaba de estado a “Calificado”, una vez que el presidente del tribunal accedía al detalle del TFG. Por otro lado, se expuso como se mostraba la evaluación final en la aplicación una vez que pinchaba en el botón calificar. Una vez calificado, se comprobó que la información quedo registrada en la tabla TFG, con la nota final del estudiante.

5.3. Pruebas de Usuario

Una parte de las pruebas consistió en la realización de test por parte de usuarios, para que puntuaran y plasmaran la experiencia de uso de la aplicación. Para ello se les pidió que navegaran por la aplicación sin darles ningún tipo de indicación y después se les pidió que realizasen un par de hitos (recogidos en el cuadro 5.1) para comprobar la facilidad de uso.

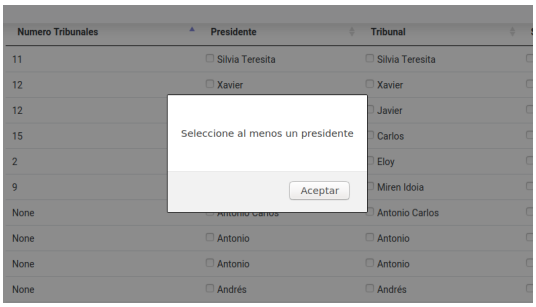
Los usuarios encuestados navegaron de forma libre por la aplicación para comprobar la manejabilidad del sistema. Entre los participantes se encontraban usuarios que conocían la plataforma y usuarios que la desconocían por completo, mostrando una amplia diferencia entre ellos. La experiencia en rasgos generales fue positiva, como se puede ver en la tabla . Al mismo tiempo sirvió para mostrar los puntos débiles de la aplicación, donde debería realizarse un mayor esfuerzo en las siguientes versiones, para que mejore la experiencia del usuario.

	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Login	15 seg	17 seg	22 seg	16 seg
Listar TFG	10 seg	8 seg	12 seg	15 seg
Seleccionar	20 seg	23 seg	12 seg	14 seg
Solicitar Defensa TFG	1 min 22 seg	1 min	1 min 15 seg	1 min 56 seg
Descargar Normativa	4 seg	6 seg	3 seg	4 seg
Examinar TFG	5 seg	4 seg	5 seg	2 seg
Subir TFG	4 seg	3 seg	2 seg	4 seg
Generar Informe TFG	59 seg	49 seg	30 seg	38 seg
Admitir TFG	2 min 22 seg	1 min 23 seg	3 min 15 seg	4 min 23seg
Hacer Publico TFG	3 seg	8 seg	6 seg	11 seg
Crear Tribunal TFG	14 seg	42 seg	34 seg	56 seg
Acceder Informe TFG	7 seg	7 seg	3 seg	4 seg
Descargar TFG	4 seg	3 seg	6 seg	8 seg
Evaluar TFG	34 seg	52 seg	24 seg	56 seg
Calificar TFG	23 seg	12 seg	41 seg	33 seg

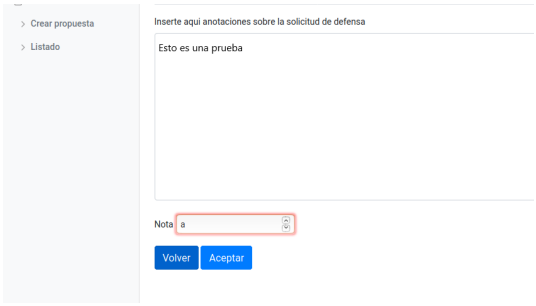
Tabla 5.1: Tabla donde quedan registrados los tiempos empleados por lo usuarios para realizar las funcionalidades del sistema.

5.4. Pruebas de Componentes

Durante el desarrollo de la aplicación se fue probando las respuestas de las funcionalidades implementadas. Cada vez que un nuevo caso era implementado se realizaba una serie de pruebas que validaba la evolución, permitiendo avanzar en el desarrollo de la aplicación. En esta sección vamos a explicar la respuesta del sistema frente a un posible comportamiento anómalo que no siga el flujo esperado de la aplicación. Un ejemplo del tipo de mensaje de error, que el usuario se encontrará mientras navega en el sistema están mostrados en las figuras figura 5.1.



(a) Notificación de error que indica que uno de los campos esta introducidos de forma incorrecta.



(b) Mensaje de error que nos muestra que el campo esta introducido con caracteres erróneos.

Figura 5.1: Muestra del comportamiento del sistema ante una entrada errónea de argumentos.

Un caso de error nos lo encontramos cuando el estudiante no selecciona un TFG para cargar. El sistema no permite avanzar de estado, además muestra en un mensaje cual es el error. Tras esto el usuario deberá seleccionar y subir el TFG deseado para permitir que el estado avance.

Lo mismo ocurre con la solicitud de la defensa, si se intenta guardar los cambios sin haber introducido ningún texto ni valor dentro de los campos. El mensaje mostrado nos recordara que es necesario completar los campos. Del mismo modo el sistema solo permite la introducción de las notas mediante caracteres numéricos, además la aplicación solo permite una única forma de introducir la nota asignada por el profesor, que es con una único decimal.

En la sección del creación del tribunal, se nos muestran diversos *inputs* para que seleccionemos que usuarios queremos que sean presidente, titulares o suplentes en la formación del tribunal. Siguiendo las normas del tribunal del TFG el sistema solo debe permitir que solo un profesor ejerza la figura del presidente del tribunal, que dos profesores formen parte como titulares dentro del tribunal y que otros 2 profesores sean los que formen parte de los suplentes. En caso contrario se mostrará un mensaje de error, explicando cual es el motivo.

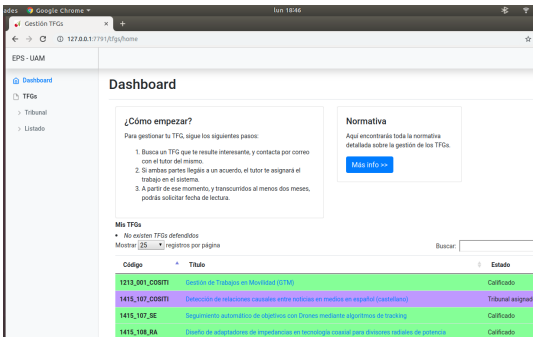
5.5. Pruebas de Compatibilidad

Una de las pruebas llevadas a cabo fue la del uso de la aplicación en los principales navegadores web, ya que este sera el acceso común a la aplicación.

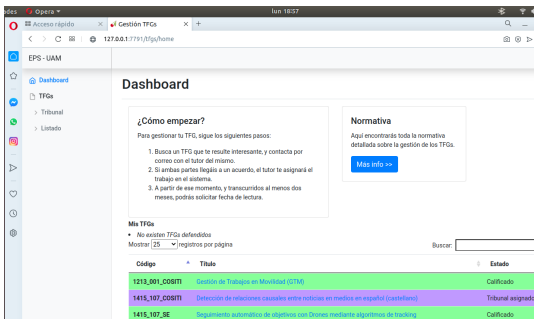
	Chrome	**Firefox**	**Opera**
Login	Correcto	Correcto	Correcto

Tabla 5.2: Esta es una tabla de ejemplo en la que, internamente, se usa el entorno **tabular**.

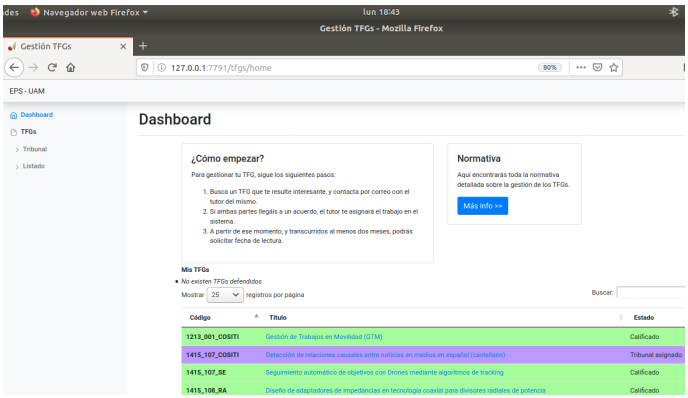
Tras los resultados mostrados, queda demostrado que la aplicación permite el acceso desde los principales navegadores web, permitiendo de este modo una mejor y mayor accesibilidad a la plataforma. Los ejemplos de esta compatibilidad se pueden comprobar en las figuras figura 5.2.



(a) Buscador web Chrome



(b) Buscador web Opera



(c) Buscador web Firefox

Figura 5.2: Navegadores probados para mostrar el correcto funcionamiento de la plataforma en ellos.

6

Conclusiones

6.1. Conclusión

La realización de este TFG supone supone la creación de una plataforma que modernice y facilite la gestion de los procedimientos relacionados con la defensa del Trabajo Fin de Grado.

Con esta premisa se ha ido desarrollando este proyecto que fue avanzando en cada fase y que termino siendo una plataforma mas potente y con mas funcionalidad de la que estaba pensada al inicio del desarrollo del trabajo. En todo momento la comunicación con los responsables del proyecto fue vital, pues esta fue fluida, y continua de nuevos conceptos y servicios interesante a desarrollar, razón por la cual se han desarrollado hasta cerca del doble de los estados que estaban previstos en la idea inicial.

No obstante el desarrollo de la aplicación ha estado sujeto a complicaciones. La instancia donde se instalo la aplicación al inicio fue quedándose obsoleta en rendimiento, lastrando el desarrollo de las funcionalidades.

La tecnología empleada en el desarrollo es ampliamente conocida y usada. El lenguaje usado para la creación de la plataforma fue impuesto por el estado de la aplicación en el momento que me incorpore al proyecto. Esta tecnología asegura un funcionamiento continuo y ligero ya que usa tecnologías y paradigmas pensados para el desarrollo de aplicaciones web livianas donde el rendimiento es primordial.

Por otro lado, la creación de esta plataforma se realizó en paralelo junto con el desarrollo de una jornada laboral completa, por lo que compaginar 2 trabajos al mismo tiempo supuso un reto. En definitiva estoy satisfecho con la deriva que llevo el desarrollo del TFG y su resultado final.

6.2. Trabajo futuro

Como todo desarrollo, hay diversas partes a mejorar, algunas de ellas fruto de falta de tiempo para dedicar una revisión y otras fruto del desconocimiento de la tecnología con la que se ha trabajo. Los puntos a mejorar son los siguientes:

Listado de tareas

Mejora 1: Una mejora interesante seria la de aumentar la capacidad comunicativa de la aplicación. Para ello se debería mejorar y activar el sistema de alertas por correo, que indica las acciones importantes que se realizan con respecto al TFG.

Mejora 2: Siguiendo con los avances comunicativos en la aplicación, propongo establecer un servicio de mensajería mediante un chat para establecer un canal comunicativo entre el estudiante y el profesor para resolver dudas.

Mejora 3: La interfaz gráfica es primordial a la hora de hacer atractiva una aplicación y facilitar el uso de esta. Propongo mejorar la visualización de los trabajos presentes en la aplicación que se muestran en la sección “Listado”.

Mejora 4: Continuando con el aspecto gráfico, planteo la mejora de la interfaz gráfica en algunas pantallas, como al mostrar la calificación del TFG o algunos mensajes de error que podrían ofrecer mas información al usuario.

Mejora 5: Por otro lado es importante centrarse en la evolución de la aplicación. Por ello sería interesante añadir y crear un nuevo estado previo a la “Solicitud defensa” que permita al estudiante compartir su TFG con el tutor para que revise el avance de este sin que implique solicitar la defensa.

Mejora 6: Por último, el alcance de este aplicación debe estar dirigido a todos los estudiantes de la UAM. Se debe establecer esta aplicación como el portal por defecto para gestionar los TFG en todas las facultades existentes en la Universidad UAM, tras pasar un periodo de pruebas en la Escuela Politécnica Superior.

Bibliografía

[1] "Trabajos fin de grado," *Escuela Politécnica Superior*, 2015. https://www.uam.es/EPS/TrabajosFinDeGrado/1446764166539.htm?, Accedido en Septiembre 2019.

[2] K. Schwaber and J. Sutherland, "The scrum guide™," pp. 3–19, 2017. , Accedido en Octubre 2019.

[3] A. Frechina, "Metodología scrum ¿que es?," *WinRed*, 2018. https://winred.es/management/metodologia-scrum-que-es/gmx-niv116-con24594.htm, Accedido en Octubre 2019.

[4] A. K. Nora Koch *et al.*, "Uml-based web engineering," *ResearchGate*, 2008. , Accedido en Junio 2020.

[5] "About sqlite," https://www.sqlite.org/about.html, Accedido en Diciembre 2019.

[6] "Acid," *Wikipedia*, 2017. https://es.wikipedia.org/wiki/ACID, Accedido en Diciembre 2019.

[7] "Acid," *Wikipedia*, 2017. https://es.wikipedia.org/wiki/ACID, Accedido en Diciembre 2019.

[8] "Why cherrypy?," https://docs.cherrypy.org/en/latest/intro.html, Accedido Mayo 2020.

[9] "Framework o librerías: ventajas y desventajas," *TiThink*, 2018. https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/, Accedido en Octubre 2019.

[10] "Compiler vs interpreter: Complete difference between compiler and interpreter," *Guru99*. https://www.guru99.com/difference-compiler-vs-interpreter.html, Accedido en Mayo 2020.

[11] I. N. Véronica Dahl, *Logic Programming*. Springer, 23 ed., 2007. Accedido en Junio 2020.

[12] "General overview of cross-platform languages," *Guru99*, 2015. https://skelia.com/articles/general-overview-of-cross-platform-languages/, Accedido en Mayo 2020.

[13] P. Giorgini *et al.*, "Goal-oriented requirement analysis for data warehouse design," *Escuela Politécnica Superior*, 2005. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.74.1396rep=rep1type=pdf, Accedido en Noviembre 2019.

[14] A. WATT, "Chapter 8 the entity relationship data model," *BCcampus*, vol. 1, no. 1, 1990. https://opentextbc.ca/dbdesign01/chapter/chapter-8-entity-relationship-model/, Accedido en Octubre 2019.

[15] M. Beedle *et al.*, "The agile manifesto," *Agile Alliance*, 2001. http://agilemanifesto.org/history.html, Accedido en Junio 2020.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID